Clustering Techniques Applied to Forensic Video Analysis

Trabajo Fin de Grado
Doble Grado En Matemáticas e Ingeniería Informática
Curso 2018–2019

Carlos Germes Ochando

Directores
Luis Javier García Villalba
Ana Lucila Sandoval Orozco

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Mayo de 2019

Acknowledgements

This work would not have been possible without the support of Luis Javier García Villalba and Ana Lucila Sandoval Orozco. I want to thank them for the support and expertise provided that greatly assisted the research. I would like to extend this gratitude to all the Group of Analysis, Security and Systems (GASS), specially to Esteban Armas Vega, Carlos Quinto Huamán and Daniel Povedano for his assistance with the atoms extraction solution, and for sharing his knowledge.

I would also like to show my gratitude to Almudena Outeda and my family and friends for all the personal support they provided me along all these years.

Contents

List of Figures

List of Tables

Abstract

As number of mobile devices arise, and most of them equipped with a camera, it becomes not only easier to generate video content, but harder to identify or classify the source of it. On this field many papers had been written, most of them exploring the PNRU noise, with good results, even if it requires a lot of computation. This work proposes an algorithm to cluster a set of videos based only on the information of the container of the video, with lower computational cost than content based approaches.

Keywords: Acquisition Source Identification, Digital Video, Forensics Analysis, Clustering, Video Container Analysis.

Resumen

Al tiempo que el número de dispositivos móviles aumenta, y con la mayor parte de ellos equipados con cámara, es cada vez más fácil generar contenido de vídeo, pero más difícil clasificar su fuente. En este tema, existe mucho trabajo previo, la mayoría explorando el ruido PRNU, con muy buenos resultados. Pese a todo, estos métodos tienen un coste computacional muy elevado. Este trabajo propone un algoritmo de clustering para agrupar un conjunto de vídeos basado únicamente en la información del contenedor del vídeo, con un coste computacional mucho menor que los basados en análisis del contenido.

Palabras clave: Identificación de la Fuente de Adquisición, Vídeo Digital, Análisis Forense, Agrupamiento, Análisis del Contenedor de Vídeo.

Chapter 1

Introduction

1.1 Motivation

As number of mobile devices arise, and most of them equipped with a camera, it becomes easier to generate video content, and it is proved as a very effective way to communicate [Bec15]. Forecasts present it as an increasing trend [Cis18]. It already represents 75% of the web traffic, and this number is thought to rise to 82% by 2021. According to Alexa ranking [Ale], YouTube is the second top site of December 2018, adding more than 300 hours of video per minute [Asl18]. Also, on the top 20 sites we can found seven social networks that include video (Facebook, Qq, Twitter, Reddit, Vk, Weibo and Instagram). Also, video surveillance is now a trend, not only at home or offices, but also to strength public security. This is the case of London, with more than 500 000 cameras installed, or the future of Chine with the "Xue Liang" ("Sharp Eyes") program.

Importance of video forensics is bounded to this phenomenon, as it also increases number of video evidences in legal matters. Nevertheless, due to improvements on computer technologies and network allows to manipulate and share video data so easily that "digital videos and photographs can be no longer considered proof of evidence, since their origin and integrity cannot be trusted" [MFB+12]. There is a need of tools to authenticate video content, trace is life cycle, identify the source, etc.

Video containers contain relevant information about the source of the file. It is important not to confuse video container information with metadata, as the former uses specific and tamper-proof information from the structure of the video file. Most part of the previous work focus on the analysis of the video content rather than the container, even if this analysis is much heavier.

1.2 Context

The current Final Degree Project is part of a research project entitled RAMSES approved by the European Commission within the Horizon 2020 Research and Innovation Framework Program (Call H2020-FCT-2015, Innovation Action, Proposal Number: 700326) and in which the GASS Group of the Department of Software Engineering and Artificial Intelligence of the Faculty of Computer Science of the Universidad Complutense de Madrid participates (Group of Analysis, Security and Systems, http://gass.ucm.es, group 910623 of the catalogue of research groups recognised by the UCM).

Apart from Universidad Complutense de Madrid, the following entities participate:

- Treelogic Telemática y Lógica Racional para la Empresa Europea SL (España)

- Ministério da Justiça (Portugal)
- University of Kent (Reino Unido)
- Centro Ricerche e Studi su Sicurezza e Criminalità (Italia)
- Fachhochschule fur Offentliche Verwaltung und Rechtspflege in Bayern (Alemania)
- Trilateral Research & Consulting LLP (Reino Unido)
- Politecnico di Milano (Italia)
- Service Public Federal Interieur (Bélgica)
- Universität des Saarlandes (Alemania)
- Dirección General de Policía - Ministerio del Interior (España)

1.3 Objectives

In this work two algorithms are proposed, capable of separating videos by encoding source, specifically by brand. To do so, the some steps were followed.

- Investigation about video forensics, and video file formats
- Design of a solution
- Evaluation of data set representation
- Evaluation of Clustering algorithms
- Interpretation of the results

1.4 Related Work

There is not many work on video container analysis, as it was not introduced until 2014 by [GFK14]. This paper introduces the information contained in MP4, MOV and AVI file formats video streams. It has no mathematical approach and it is limited to state the differences between devices, and notice its potential for video forensics purposes.

This first work, still on very early stages, was developed by [SPC17] resulting in the paper [ISF+19], aiming at video integrity verification and identification and classification of the source. This last paper was the reference of this work, as it is the most developed production on this topic.

Even if video container had never been used for forensic purposes before, a large amount of work exists about video forensics. An overview of them is given later.

1.5 Structure

The rest of this work is divided into 5 more chapters:

Chapter 2 begins with an introduction to how video files work, and continues with an overview of forensic video analysis. Based on the main concepts, some of the techniques developed to cover its three main areas are detailed: Acquisition, Compression and Video Compression. However, none of these techniques is related to video containers. From this

point he focuses on the state of the art of video container analysis. The work of [ISF+19] is the main theme of this chapter, since it is one of the few works on the subject, and the most recent and advanced. This work is analysed and will be the basis of the next chapter.

The chapter 3 contains the work proposals. It explains how to reformulating the base of the work in [ISF+19] grouping techniques can be applied. It also details the techniques used. One important step is the choice of the representation of the videos. These different ways of approaching the data are the core of this work, since the adequate representation of the data is crucial for the grouping without supervision. Finally, two algorithms are proposed to perform the grouping.

The chapter 4 describes the experiments carried out and their results. It is divided into two phases. It begins with an analysis of the data set in different different representations. For this we measure the silhouette coefficient of the different representations. Finally, the two grouping algorithms are tested using the most promising representations.

Chapter 5 concludes this work, summarizes it and defines possible improvements or extensions that could be made in the future.

Chapter 2

Video Forensics

The forensic analysis of video containers had never been discuss before 2014 [GFK14], and first mathematical came 4 years later [SPC17], as part of a thesis. This chapter analyses the result of their work, and overall the approach they took to the data.

2.1 Video File

The following chapter peeks at digitisation of video content. Since the XIXth century, video had been recorded in analogical support, like film, but from late XXth century, with the uprising of computers, digital video has became a daily reality.

A video is just images along time, and maybe also audio. So there is a need of a new file format able to manage this multimedia streams: video file formats. It is also important to talk about video encoding, which is very related with compression. Imagining video data as a collection of images to be displayed, and using uncompressed images, the amount of information needed per second is huge. The formula is quite simple:

$$\text{bit rate (bps)} = \text{colour depth (bits)} \times \text{vertical resolution} \times \text{horizontal resolution} \times \text{refresh frequency (fps)}$$

Then doing easy maths, a depth colour of 8 bits (256 colours, VGA palette) and a resolution of 720x576 and 25fps, similar to European TV PAL resolution, it will have a bit rate of 83 Mbps (10 MBps). For example, the original “Star Wars: Episode IV – A New Hope“ (121 min) will have a size of 75GB in this quality. Speaking about 1080p 60Hz, which correspond to a Full HD television, it will be up to 1Gbps and almost 1TB for the movie. And this is without accounting audio tracks!

2.1.1 Video File Formats

There is a common confusion in the concepts of video file, due to undistinguished use of file formats and video codecs when they designate two very different concepts. A file format, container or wrapper ”describes how different elements of data co-exists within the file” [Sci17]. Figure 2.1 is a simplified representation of video file (or container or wrapper).

Each container may allow different video or audio stream types and numbers, Table 2.1 shows some of the most common file formats that can be found in real world. The two last ones, QuickTime and MP4 are implement an ISO standard (inspired by the Apple

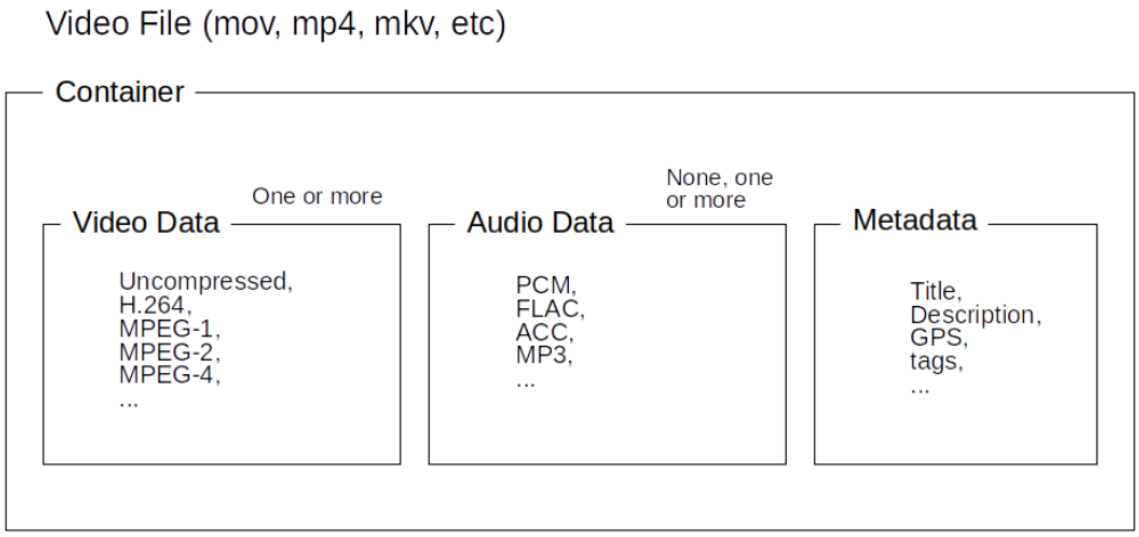


Figure 2.1: Simplified view of a video file

format), ISO/IEC 14496-12 [Int17], thus, will have a similar format, that will be reviewed in depth in further sections.

Name	Extension	Container	Video Encoding	Audio Encoding
AVI	.avi	AVI	Any	Any
FLV	.flv	FLV	H.264, VP6, ...	AAC, MP3, ...
Windows Media Video	.wmv	ASF	Windows Media Video, ...	Windows Media Audio, ...
QuickTime	.mov	QuickTime	Apple Video, H.262, H.264, ...	AAC, MP3, ...
MPEG-4 Part 14	.mp4	MPEG-4 Part 12	H.264, MPEG-4 Part 2, ...	ACC, MP, ...

Table 2.1: Some of the most popular video file formats

2.1.2 Video Codecs

That is why video codecs exists. A video codec is an "algorithm" able to code and decode a signal, a video in this case. To reduce the size of a file, redundant information is removed, this is called lossless encoding, but this has a limit. Once crossed, it start to drop information, and it is called lossy encoding. Lossy encoding might introduce artifacts on the video, but these might be negligible or tolerable. As videos are images along time, it naturally produces two kinds of compression: spatial and temporal.

Spatial compression is the heritage of image codecs, as every frame is compressed as an independent image. For example, JPEG image codec uses divides each frame in blocks of pixels, then performs a chroma subsampling, transforms them using discrete cosine transform (DCT), and at last quantization on the resulting coefficients to reduce images sizes at 10:1 without visible effect for human eye [nas92]. These techniques are also adopted by MPEG-1, MPEG-2 and current MPEG-4 [Int15].

Temporal compression tries to reduce redundant information in different frames. Think of a static movie take, with only one object moving across. Only this object is interesting, as the background is static in all the takes. To do so, the concept of Group Of Frames (GOP) is introduced. Specifications may vary among different codecs, but in general general line the schema is as follows:

- **I-frames:** known as key frames or reference frames are not temporally compressed, so they can be "read" alone.
- **P-frames:** known as predictive frames, they are rendered using previous I-frame information, or they can be a standalone frame (like an I-frame) if changes are too big.
- **B-frames:** known as bidirectional frames, they reference the nearest I-/P-frames forwards and backwards..

This is a general approach, for example H.264, allow B-frame to be used as reference for P- and B- frames. This has pros and cons, as it can save space by not repeating changes already done on previous B-frame, but penalise low quality B-frames as artifacts will be propagated. Or MPEG-2, that imposes P-frames to only reference the single previous I-frame while H.264 allows multiple references without concern of being the previous.

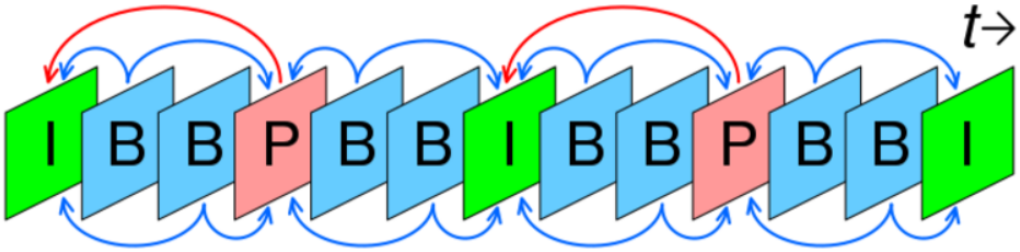


Figure 2.2: Representation of two group of frames. Arrows indicate dependencies to rebuild the frame

The pattern of the GOP is determined by a sequence starting with an I-frame and the following P- and B-frames before the next I-frame. In the example of Figure 2.2 the sequence is IBBPBB. The length of this sequence, called GOP length, is the measure of frames between two I-frames. The bigger, the most compressed the video will be, but fast transitions will lose quality. The compromise is achieved depending on the situation: video conferences will surely use longer GOP lengths than movies (like action movies).

2.2 Concept of Video Forensics

Three main areas may be distinguish on digital video signals forensics: acquisition, compression and edition.

The first concerns the first steps of the content, which is usually translated in the identification of the source of originating device. The second aims to determine the life cycle of the content, usually translated in multiple compression of an original video content. The last one attempts to determine if video content as suffered some kind of alterations.

But before diving in these areas some terms must be clarified, in order to clearly understand the nuances:

- **Integrity:** " ensuring that the information presented is complete and unaltered from the time of acquisition" [oDE16].
- **Authentication:** "The process of substantiating than the data is an accurate representation of what it purports to be" [oDE16].

- **Classification:** "assign the most likely source brand from a set of known brands" [ISF+19].
- **Identification:** "checking the compatibility degree of a query video with an alleged source brand" [ISF+19].

The first two terms seem very similar in current English, but it is clear that the integrity is a stronger condition, as any authentication violation will also compromise integrity. Integrity is, nevertheless, sometimes negligible the integrity of a video: in real world cases, video will usually suffer for re-compression processes (any sharing on social networks will cause it) probably without compromising its authenticity.

As integrity is a very restrictive condition, very bounded with the origin source, it is often needed to first asses this to have a comparison point. In order to do that a video can be classified among a set of well known brands, or checked against video from supposed source.

2.2.1 Acquisition

This is the main area intend to be covered in this work. The objective been to determine which is the exact device was the generator, it can be divided it in 3 questions: the type (CCVT, smartphone, etc), the model (brand and/or model), and then the exact device (determine if a specific device is the responsible). Each of this steps could be relevant by its own, as it might supply crucial information.

There have been developed many approaches using the analysis of the noise in the video, being the PNRU the most significant because of his robustness, to extract a fingerprint of the camera device that recorded it. This has been proven effective for images, but is a challenging task for video due to lossy compression 2.2.2 it is often performed. There have been nevertheless positive results for identifying the PNRU noise of a device even on heavily compressed video form social networks as shown in [ACDM+17].

As it may have been noticed, most of this techniques are focused on the video stream analysis, which is translated in huge volumes of information that need to be manipulated.

Added to this, thanks to the optimisation of chips manufacturing, it is possible to manipulate and alter this videos content from a simple entry level smartphone. Not only that but social networks, as WhatsApp, Facebook, but also Youtube and so, making almost any video potentially unreliable.

2.2.2 Compression

As stated before, uncompressed video needs from huge amounts of space, that is why when a video is captured (or generated in any way) it is usually encoded at the moment. As the spectrum of file formats and encoding is so ample, each device/model/brand may use different encodings, and thus, identifying video encoding parameters might be useful not only trace how the video was generated or even to estimate the quality of the video without having access to to source.

It was seen in 2.2.1 it is prevailing the re-encoding of video when uploaded to social networks of any kind, but it also happens when a video is edited in any way. Then, the study of the life cycle of the video may contribute to identify not only the original format but to trace this history too.

This history of re-encoding can be helpful to trace its route, even up to the adquisition device.

Many of videos are loosely encoded, for example any MOV or MP4 video. This loosely compression is very characteristic, and as a specific footprint. Some of the most common techniques for compression do focus on detecting the size of the blocks used for compression, and others on profiling the parameters of the quantization.

Those techniques are the same used for JPEG, as the spacial compresion on both is equivalent. On both techniques, the job of Z Fan, RL De Queiroz et al overstands [FDQ00] [FDQ03].

Another techniques had been developed to detect double compression, that is when an already loosely encoded video is encoded again. Due to quantization, the histogram of the DCT coefficients produces peaks, that can be exploited, as shown in [HLWT06]. Another approach is also based on DCT coefficients, and Bendford's law, stating that a double encoded video will break it, as in [FSS07].

Other techniques are focused on the temporal compression, as in [VTT10].

2.2.3 Edition

Last area is the detection of video tampering, which affects the authenticity of the video. This might seem easier for video than for images, as it looks more complicated to edit a video than an single frame, but removing or duplicating is a very simple task that does not introduce any evidence of tampering in the individual frames.

To detect manipulations in video, there have been different approaches adopted. The more intuitive one is an heritage from image authentication [Sci18], based on the analysis of the inconsistencies of "what it is seen", as incoherent shadows, but this will nevertheless be a very complex task. This did not prevented some works in this direction, identifying inpainting [ZSZ09], or analysing free-flight movement objects [COF12] to detect manipulations.

Another approach is to trace the camera noise among the different frames and regions. Reusing techniques used for acquisition identification of the source [MCP+07], allows to profile the PRNU noise of each frame to look for correlation on the video. Knowledge on this techniques also lead to the analysis of other noises, as "noise residue" [HHLH08]. Using denoising filters developed for the calculus of the PNRU [MKR99], extracting this noise and getting its correlation between temporally adjacent blocks. Repeated frames will rise correlation to 1, and edited ones will decrease this value, revealing and even locating the tampered frames and pixels. It has been proven effective in uncompressed videos and promising in compressed ones.

When a video is modified it will probably need to be re-encoded (or maybe not, depending on the nature of the modification), and so, even if it might difficult the use of previous techniques, it leaves a trace, as seen in section 2.2.2. For example, when re-encoding a video, different GOP may arise and this will, with lossy encoding as MPEG, have consequences in the P frames encoding [WF06], revealing frame deletions or additions, or even doctoring, if only part of the video suffer this phenomena, or the video is not supposed to have been re-encoded. Another approach tries to detect double compression per block looking for MPEG (JPEG) quantization artifacts [WF09], allowing to detect macro-blocks (16x16 pixels when MPEG standard is 8x8) doubly compressed. Applied frame wise this allows to detect green screening in much cases if the quality of the background is lower enough compared to the resulting video.

2.3 Forensic Analysis of Video File Formats

The paper [GFK14] offers a glance at metadata contained on AVI and MOV-Based files. It is just a first approach at the information contained in both wrappers, and how it varies among recording devices. They use a dataset with various smartphones, cameras, and even video editors.

2.3.1 Files Structure

MOV-Based (MP4 like) and AVI do have a similar structure for metadata: while MP4-like rely on "atoms", and AVI on "lists" and "junk", both present a hierarchical structure. The standard specifications however differ: AVI is much more relaxed than MP4, and that affects on how to use the information.

2.3.2 AVI Files

For the analysis of the AVI files the focus is on the structure of the container more than the values. Once covered the mandatory header, the structure of the file is unrestricted, and thus, the structure defined by "lists" and"junk" is even more characteristic than the values they may have.

The authors are even able to identify unique specifics for some models over they dataset. This characteristics are potentially suitable to classify as produced by a specific camera or editor, or at least to discard them.

2.3.3 MOV Based Files

For MOV based files, the structure is more restricted than for AVI. Nevertheless, even if better defined, it is not completly fixed. It is still insteresting to analyse the structure of the atoms forming the video, noticing some particularities. For example, even if usually MP4 videos start with ftyp atom, Kodak ones start with skip atoms.

The values itself contained in the structure reveal an interest too. The authors start with the focus on ftyp fields "Major brand" and "Compatible brands" with fixed values for each class of their dataset. Even more, almost each class as its own combination of values as shown in Table 2.2, if they have. So the ftyp information seem to be a good starting point for video classification.

Furthermore, video encoding information reveal potential too. Authors point out "time_scale" parameter of video and audio streams, representing the frame rate and audio sampling rate, as an example of the relevant information that can be found in the moov atom. This atom is the one that contains the information to decode the data stream contained in mdata, and contains specific information on video and audio encoding used.

Model	Major brand	Compativle brands
Apple iPhone 4	qt	qt
Benq S88	isom	isom, 3g2a
BlackBerry 8310, Palm pre	3gp4	3gp5, 3gp4, isom
Canon 7D	qt	qt, CAEP
Google Nexus 7	3gp4	isom, 3gp4
Kodak M1063	-	-
LG J990	3gp5	3gp5, 3gp4
Samsung SGH-D600	3gp5	3gp5, isom
Minolta DiMAGE Zq	-	-
Motorola MileStone, Samsung GT-5500i (MP4V)	3gp4	3gp4, mp41, 3gp6
Samsung GT-550i (H.263)	3gp4	3gp4, 3gp6
3GP: Nokia 6710, E61i, E65	3gp4	3gp4, 3g2a, isom
MP4: Nokia 6719, E61i, E65	mp42	mp42, 3gp4, isom
Nokia X3-00	3gp5	3gp5, 3gp4, 3g2a, isom
Praktica DC2070	-	-
SonyEricsson K800i	3gp5	vfj1, 3gp4, 3gp5, mp42
FFmpeg	isom	isom, iso2, mp41
YAMB	mp42	isom, mp42, 3gp5
Adobe Premiere CS5	3gp5	isom, 3gp3, mp41, mp42

Table 2.2: Major and compatible brands stored in the MP4 ftyp atom [GFK14]

The paper states that different video sources produce characteristics footprints in the video file structure (both AVI and MOV based) that can exploited.

It is assumed that the characteristics manually spot are not enough to define a model. Then, as furthers steps, it is suggested to check how this can scale in larger datasets, where unknown file formats may appear, and use parsers to automate information extraction.

2.4 A Video Forensic Framework for the Unsupervised Analysis of MP4-Like File Container

The paper [ISF+19] is focused only on MOV-based file, and a more automatised approach is attempted. This paper is, nowadays, the state-of-the-art on video container analysis.

2.4.1 Extraction and Preprocessing of Video Files

The authors use a MP4 parser [APA18] to extract atoms information from MP4-like video. Then, observations X are build like a collection of field-values ω with an associated path and order $p_X(\omega)$. For example,

$$X = (\omega_1, ..., \omega_m) \tag{2.1}$$

with filed-values and paths like

$$\omega = @creationTime : 2017 - 07 - 2809 : 14 : 13$$

$$p_X(\omega) = (moov - 4, mvhd - 1)$$

Once defined the shape of the data, the authors define some formulas that will be useful to work with this data:

- **Similarity between field-values**
  Given two videos $X = (\omega_1, ..., \omega_m)$ and $Y = (\omega'_1, ..., \omega'_n)$ similarity between field-values is defined as if two field-values have the same value and path.

$$S(\omega_i, \omega'_j) = \begin{cases} 1 & \text{if } \omega_i = \omega'_j \text{ and } p_X(\omega_i) = p_Y(\omega'_j) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

- **Comparison between field-value and video**
  Given two videos $X = (\omega_1, ..., \omega_m)$ and $Y = (\omega'_1, ..., \omega'_n)$ the comparison of a single $\omega_i \in X$ with a whole other video Y as:

$$1_{X'}(\omega_i) = \begin{cases} 1 & \text{if } \exists \omega'_j \in Y : S(\omega_i, \omega'_j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

  This can be read as 1 if Y contains an attribute similar to ω_i.

2.4.2 Video Integrity Verification

Using the structures defined the authors define a metric-like formula to measure dissimilarity between two videos. Then, this measure is used to compare original videos from a specific device between them, and a supposed original video. If the maximum distance between well-known original videos is smaller than the maximum distance from the unknown video to the original ones, the video is called tampered.

The mismatching percentage of all field-values is:

$$mm(X, X') = 1 - \frac{\sum_{\omega \in X} 1_{X'}(\omega)}{card(X)} \tag{2.4}$$

This formula is not symmetric as it is then percentage of field-values in X with a similar one in X' but if the cardinality is different or there are repeated values (it is possible) it will affect the results. In order to compensate, dissimilarity is defined as the mean of both values:

$$D(X, X') = \frac{mm(X, X') + mm(X', X)}{2} \tag{2.5}$$

2.4.3 Brand Identification and Classification

In this section the authors introduce the concept of Likelihood ratio applied to videos. The so called likelihood ratio on shullanivideo and [ISF+19] is in fact the Bayes factor of two hypothesis given a video $X = \{\omega_1, ...\omega_m\}$:

$$H_0 : X \text{ belongs to } \overline{C} \tag{2.6}$$

$$H_1 : X \text{ belongs to } C \tag{2.7}$$

This approach has been proven effective for DNA analysis and accepted in court [BS15], and also called weight of evidence.

First, they denote by Ω the set of "distinct" field-values (using the formula $S(\bullet, \bullet)$) present on the universe of all videos $\mathcal{X}$.

They also define, given a class $C \in \mathcal{C}$, the subset of videos belonging as $\mathcal{X}_C$, and his complementary $\mathcal{X}_{\overline{C}} = \mathcal{X} \setminus \mathcal{X}_C$

Now the "discrimination power" each $\omega \in \Omega$ can be determined, for this class C:

$$W_C(\omega) = \frac{\sum\limits_{X_i \in \mathcal{X}_C} 1_{X_i}(\omega)}{card(\mathcal{X}_C)} \tag{2.8}$$

$$W_{\overline{C}}(\omega) = \frac{\sum\limits_{X_i \in \mathcal{X}_{\overline{C}}} 1_{X_i}(\omega)}{card(\mathcal{X}_{\overline{C}})} \tag{2.9}$$

This value $W_C(\omega)$ is just the percentage of videos of the subset containing the field-value. It is time to pose the key point of this approach.

Now, the core is to approximate for each $\omega \in X$ the conditional probabilities:

$$P(\omega|H_0) = W_{\overline{C}}(\omega)$$
$$P(\omega|H_1) = W_C(\omega)$$

This approximation allows them to easily calculate the likelihood ratio (Bayes factor) of each field value as:

$$L_C(\omega) = \frac{P(\omega|H_1)}{P(\omega|H_0)} = \frac{W_C(\omega)}{W_{\overline{C}}(\omega)}$$

Assuming each ω_i is independent in the vector $X = \{\omega_1, ...\omega_m\}$, they get:

$$P(X|H_k) = \prod_{\omega_j \in X} P(\omega_j|H_k)$$

With the assumption and using previous approximation, likelihood ratio of a video X and a class C is:

$$K_C(X) = \frac{P(X|H_1)}{P(X|H_0)} = \prod_{\omega_j \in X} L_C(\omega_j) \tag{2.10}$$

In order to get a more suitable scale, the log of $K_C(X)$ will be called likelihood ratio:

$$L_C(X) = log \prod_{\omega_j \in X} L_C(\omega_j) \tag{2.11}$$

Using this formula an array assigning one ratio to each class $C_j \in \mathcal{C}$ can be computed, getting a function $X \to (L_{C1}(X), \ldots, L_{Cs}(X))$. The interpretation of the ratio may vary, depending on the evaluation method. The authors suggest to consider H_1 if $L_C(X) > 0$. This approach can result in multiple classified sources, as multiple classes can give a positive likelihood ratios. They then opt for the highest $L_C(X)$.

As mentioned previously formula also lays in the assumption of the independence of $\omega_j \in X$ which is not guaranteed as they might be repeated field-values in the same path, or having correlations between fields. This will produce biased ratios, by accounting multiple times the same information.

In order to mitigate the effect of correlated values, authors suggest to insert a decorrelation factor to formula 2.11 used to compute the likelihood ratio of video $X =$

$(\omega_1, ..., \omega_n)$ is modified applying an exponent α_i to each $L_C(\omega_i)$ based on the repetitions of ω_i in the video.

The approach is however useless to deal with correlations between fields values, and was not even tested on the experimental phase.

In this paper the authors state a first approach on automated video classification and video integrity for MP4-like videos, with promising results. This approach lacks, however, of a more refined way of comparing field-values to reflect the expected changes between videos. They also assume that the measure used to compare videos, the dissimilarity, lacks robustness, and should be reviewed.

Chapter 3

Container representation and video clustering

This chapter will describe what has been achieved related to the objective set up on chapter 1.

3.1 Overview

The current work has been focused on MP4-like file formats, these are formats compliant with ISO/IEC 14496 Part 12 [Int17], or based on Apple QuickTime [App16] standard, so it covers not only MP4 but also MOV files. As shown in [GFK14], both standards define the video file as a sequence of atoms. The atoms types are also specified by the standard, and are composed by a header (identifying the type) and a data box that may include field-value (leaves) or other atoms (like sub-trees), generating a kind of hierarchical tree. However, this specification is open to different implementations, and so, discrepancies on the structure appear depending on the creator of the file. Video and audio encoding parameters also differ, making video container not so standard. This degree of freedom was proven useful in [ISF+19] for integrity verification and source identification.

The scope of this work was to extend the work of [ISF+19] and define a clustering algorithm to group video files by source without supervision. The aim should be to split by the exact device, but as the container depends mainly on the software, and just a bit on hardware specifications, this has not been possible. The granularity is then settled for brand grouping. On the contrary, the clustering of videos from social networks will be added to the scope.

Two solutions will be proposed, relying on similar basis, to tackle this problem. None of both is perfect, but both can help at splitting videos for further analysis. This approach is not intended to replace supervised algorithms for identification of the source, but to complement the analysis of video files.

3.2 Technologies

Some tools had been used in this work for two main purposes: to extract data from video files, and to manipulate the extracted data. The full code can be found in `https://gitlab.fdi.ucm.es/esteban/atoms-project-tfg/tree/cgermes_tfg`.

3.2.1 Information Extraction and Storage

To extract the atoms information, a solution written in Python and developed by the Group of Analysis, Security and Systems (GASS) of the department of Software Engineering and Artificial Intelligence at Universidad Complutense de Madrid has been used. This solution is able to parse multiple information from any MP4/H.264 video, MOV video, and 3gp, to extract the atoms information. The values inside had been converted to utf-8 when possible and if not it is kept as hexadecimal. The solution was modified to extract some more information and split it by components, as explained further.

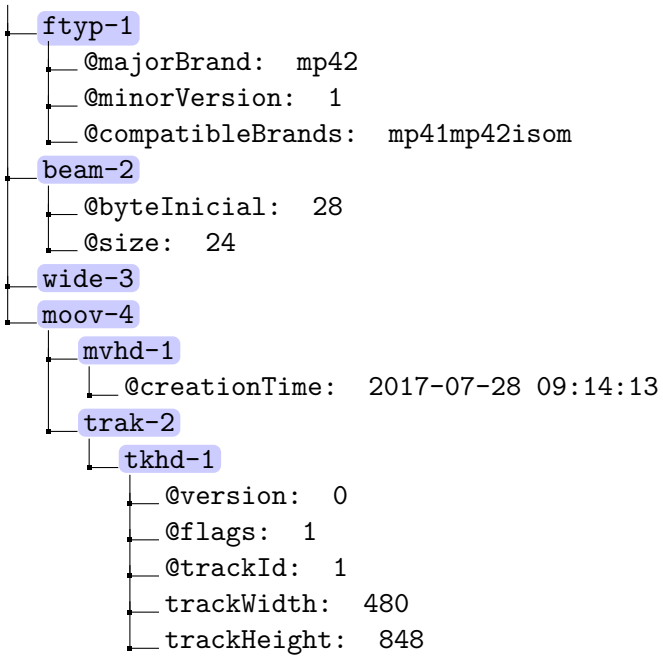


Figure 3.1: Partial example of a Samsung Galaxy S3 mini video

The Figure 3.1 is an example of how the information of the video container looks like. Atoms are marked in blue and field-values with a '@'. Only part is represented as the specific video used has 11 atoms and 55 field-values. This example shows the *ftyp* atom, which was one of the first analysed in [GFK14] as "the file type atom identifies the file type specifications with which the file is compatible" [App16]. It contains, for example, fields @*majorBrand* and @*compatibleBrands*, to identify the a more specific version of the implementation of the file. As stated before, there are many under the umbrella of the ISO standard.

From this information from the video, for each field-value, the solution gets the Path, the PathOrder which adds the relative order from its siblings, the Field and the Value. For the given example in Table 3.1, the output is shown in Table 3.1. For our purposes, videos are defined as sets of PathOrderField, that is: the PathOrder and the Field pasted together with a "/" as separator. The result is also shown in the same table.

Path	PathOrder	Field	Value	PathOrderField	PathFieldValue
ftyp	ftyp-1	majorBrand	mp42	ftyp-1/majorBrand	ftyp/majorBrand=mp42
ftyp	ftyp-1	minorVersion	1	ftyp-1/minorVersion	ftyp/minorVersion=1
ftyp	ftyp-1	compatibleBrands	mp41mp42isom	ftyp-1/compatibleBrands	ftyp/compatibleBrands=mp41mp42isom
beam	beam-2	byteInitial	28	beam-2/byteInitial	beam/byteInitial=28
beam	beam-2	size	24	beam-2/size	beam/size=24
moov/mvhd	moov-4/mvhd-1	creationTime	2017-07-28 9:14:13	moov-4/mvhd-1/creationTime	moov/mvhd/creationTime=28/07/2017 9:14:13
moov/trak/tkhd	moov-4/trak-2/tkhd-1	version	0	moov-4/trak-2/tkhd-1/version	moov/trak/tkhd/version=0
moov/trak/tkhd	moov-4/trak-2/tkhd-1	flags	1	moov-4/trak-2/tkhd-1/flags	moov/trak/tkhd/flags=1
moov/trak/tkhd	moov-4/trak-2/tkhd-1	trackId	1	moov-4/trak-2/tkhd-1/trackId	moov/trak/tkhd/trackId=1
moov/trak/tkhd	moov-4/trak-2/tkhd-1	trackWidth	48	moov-4/trak-2/tkhd-1/trackWidth	moov/trak/tkhd/trackWidth=48
moov/trak/tkhd	moov-4/trak-2/tkhd-1	trackHeight	848	moov-4/trak-2/tkhd-1/trackHeight	moov/trak/tkhd/trackHeight=848

Table 3.1: Table with Path, PathOrder, Field and Value of the elements shown in Figure 3.1, alongside with the corresponding PathOrderField and PathFieldValue

For the shake of efficiency and reproducibility, the extracted information has been uploaded to a mongoDB in "videoforensics-kzcri.gcp.mongodb.net/test" that can be read using the username "public" without password. Therefore, it is available to the public.

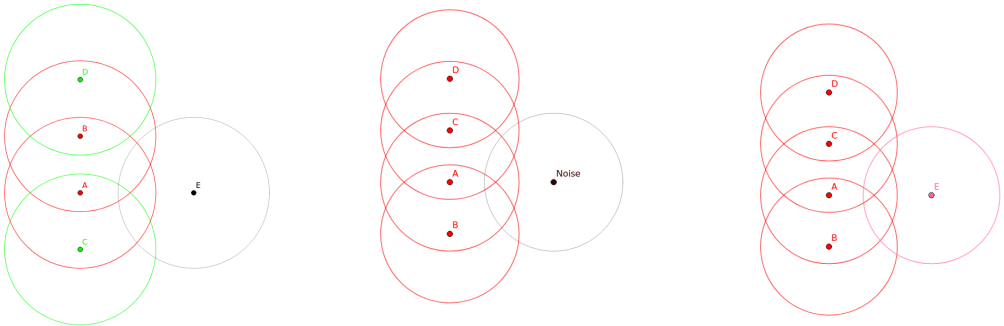
3.2.2 Preprocessing and Data Manipulation

This part had been developed in Python too, using Jupyter notebooks. The libraries used include pymongo to download the data from the database, pandas to structure it, imblearn for oversampling, and many sklearn and scipy modules. From sklearn CountVectorizer to vectorize the lists of elements, DBSCAN and metrics. From scipy, metrics, hierarchy clusters. Additionnaly, many minor libraries had been used, as classic matplotlib and Collections.

3.2.3 DBSCAN

Density-Based Spatial Clustering of Applications with Noise, proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander y Xiaowei Xu in 1996 [EKSX96], is a well-known clustering algorithm.

The algorithm tries to generate clusters based on distances between point. Each cluster is composed by at least one core point, and its edge. One point is core point if it has at least $minPoints$ within distance ϵ (including itself). Then, all edge points are those within distance ϵ from a core point. In other case, the point is considered noise.



(a) $minPoints = 3$ (b) $minPoints = 2$ (c) $minPoints = 1$

Figure 3.2: Examples of DBSCAN with variations of $minPoints$, with core points in red, border points in green, and noise in black

In this brief description appeared the two parameters of the algorithm: ϵ and $minPoints$. Both are related with the density of the clusters, however the last one is more confusing. A high $minPoints$ value indicates not only that no cluster can be smaller than it, but also that density must be high to allow core point to have at least $minPoints$ close neighbours.

The example in Figure 3.2 shows the effect of the variations in the $minPoints$ parameter. With $minPoints = 3$ the first figure shows 2 core points (red), 2 on the border (green) and 1 noise point (black). As the $minPoints$ decreases the number of core point increases until a new cluster arises with $minPoints = 1$. If $minPoints > 3$ then every point will be noise.

3.2.4 Hierarchical Clustering (Agglomerative)

Hierarchical clustering is based on distances between point, creating a tree with the points on the leafs, and the nodes been groups of videos. The joining of two nodes is based on a metric, and a linkage method.

Linkages define how the distance between two nodes is computed, based on the metric selected. When nodes are leafs (points), for every linkage, the distance between them is simply the distance between the points. When a node is composed of multiple leafs, thinks get more complex. The simpler linkages are the minimum distance between every pair of points from each node, and the maximum one. More complex linkages take the average. Linkage criterion selected for the proposed solution is the weighted average, defined recursively as:

$$d((a \cup b), c) = \frac{d(a, c) + d(b, c)}{2} \tag{3.1}$$

The Table 3.2 sumarised the criteria. Criteria used in the proposed algorithm is the weighted average.

Linkage	**Formula**
Single	$min\{d(a, b) : a \in A, b \in B\}$
Complete	$max\{d(a, b) : a \in A, b \in B\}$
Unweighted Avg	$\frac{1}{\lvert A\rvert \lvert B\rvert} \sum_{a \in A} \sum_{b \in B} d(a, b)$
Weighted Avg	$d((a \cup b), c) = \frac{d(a,c)+d(b,c)}{2}$

Table 3.2: Common linkage criterion for hierarchical clustering

The result of the linkage can be summarise in a dendrogram, like the shown in Figure 3.3. Each node represents the linking of two nodes at a height h.

Then, when the linkage is done and the hierarchy tree is defined, it remains how to extract the clusters. The method used is to cut a node when the height h is unusually high. The inconsistency is, shown in , a measure of how unexpectedly high is the height of the node. Other possibilities, like preserving only the top N clusters, require the number of clusters to be known. A node is cut when its inconsistency is higher than a threshold t. The inconsistency is calculated based on average and standard deviation of the heights of its lower level nodes in a depth d.

$$inconsistency = \frac{h - average}{standard\_deviation} \tag{3.2}$$

(3.3)

Where *average* and *standard_deviation* are computed over all the height of the descendants of the node up to a depth *depth*.

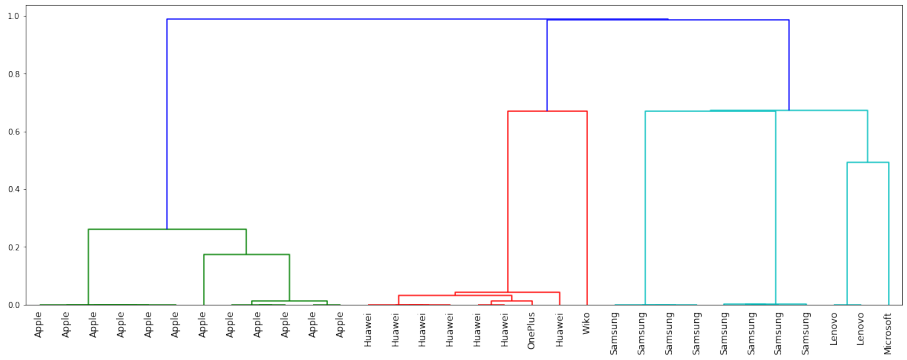


Figure 3.3: Example of dendrogram

3.3 Description of the Solution

The first solution will use Hierarchical Clustering and inconsistency criteria to cut down the different clusters, ensuring that each cluster is composed exclusively by videos of one single brand.

The second solution rely on DBSCAN to preform a clustering either homogeneous nor complete but still accurate of the videos by source.

- **Extraction of the data**
  The solution described in 3.2.1 extracts from MP4-like videos the container information split as specified, and uploads it to a mongoDB.


```
{
    "FileName":"D17_V_flat_move_0001.mp4",
    "Source":"Original",
    "Device":"D17",
    "Brand":"Microsoft",
    "Model":"Lumia640LTE",
    "Scenario":"flat",
    "Type":"move",
    "atoms":
        [
        {"Field":"order",
         "Path":"/ftyp/",
         "PathAndOrder":"/ftyp-1/",
         "Value":"1"
        },
        {"Field":"majorBrand",
         "Path":"/ftyp/",
         "PathAndOrder":"/ftyp-1/",
         "Value":"mp42"
        },
        ...
        ]
}
```


Figure 3.4: Example of the representation of the data extracted from videos with the API described in 3.2.1

- **Gather the data**
  With the information gathered, query the database to download the requiered videos to cluster alongside with the array of elements composing the video, PathOrderField or PathFieldValue, and recover the data in a dataframe. A sample of the resulting dataframe for PathOrderField is shown in Table 3.3.

FileName	Element
D01_V_outdoor_panrot_0002.mp4	/moov-3/mvhd-1/currentTime
D01_V_outdoor_panrot_0002.mp4	/moov-4/trak-2/tkhd-1/version
D01_V_outdoor_panrot_0002.mp4	/moov-4/trak-2/tkhd-1/layer
D01_V_flatWA_still_0001.mp4	/moov-4/trak-3/mdia-3/minf-3/dinf-2/dref-1/url
D01_V_indoorWA_still_0002.mp4	/moov-4/mvhd-1/timeScale

Table 3.3: Random example of the gathered data

- **Vectorization**
  Using pandas group by video concatenating each element composing the videos delimited by double quotes ("). This middle stage is shwon in Table 3.4. Thanks to the group by the CountVectorizer (a text mining tool) can be used to generate a matrix where each row is a video, each column an element, and 0/1 indicates the video having this element (not the actual count of appearances). The result is shown in Table 3.5. The benefits of vectorizing is the simplification of the further manipulation of the data. For example, the dissimilarity formula introduced in [ISF+19] is reduced to a simple vectors operation, as described in 4.2.2.

FileName	Element
D01_V_flatYT_still_0001.mp4	"/ftyp-1/majorBrand", "/ftyp-1/minorVersion", ...
D01_V_indoorYT_panrot_0002.mp4	"/ftyp-1/majorBrand", "/ftyp-1/minorVersion", ...
D01_V_indoorWA_panrot_0001.mp4	"/ftyp-1/majorBrand", "/ftyp-1/minorVersion", ...
D01_V_flatWA_still_0001.mp4	"/ftyp-1/majorBrand", "/ftyp-1/minorVersion", ...
D01_V_indoor_move_0002.mp4	"/ftyp-1/majorBrand", "/ftyp-1/minorVersion", ...

Table 3.4: Random example of the data grouped by video file

FileName	/beam-2/notEspecified	/ftyp-1/compatibleBrands	/ftyp-1/majorBrand	/ftyp-1/minorVersion	/moov-3/mvhd-1/creationTime
D01_V_flat_move_0001.mp4	0	1	1	1	1
D01_V_outdoorWA_panrot_0001.mp4	1	1	1	1	0
D01_V_flatYT_move_0002.mp4	0	1	1	1	0
D01_V_outdoorWA_panrot_0002.mp4	1	1	1	1	0
D01_V_indoorYT_panrot_0001.mp4	0	1	1	1	0

Table 3.5: Part of the dataframe after vectorization

- **Filtering**
  Using pandas, filter the columns corresponding to the unwanted atoms and fields. The removed elements are those specified in 4.2.3: the ones containing *mdta* and *udta* in the path (with or without order) and the ones with field *uuid*, *payload*, $notEspecified$, *xyz*, *entries*, *size*, $modificationTime$, $stuff$, $creationTime$, *duration*, *entryCount*, *sampleCount*, *unkn* or $freeSpace$. This filtering is an extension of the one performed in [ISF+19].

Even if the columns names are strings, thanks to the uniform format and regexp their are easy to spot. The output is exactly alike the one in Table 3.5 but without the concerned columns.

- **Clustering data**
  Recover the binary matrix of values from the dataframe representing your dataset. Each line represents an observation, video file, and columns dimensions. To cluster the data, two algorithms are proposed: For PathFieldValue, use Hierarchical Clustering and dice metric and cut the node when the inconsistency value, calculated for a depth of 2, exceeds 1,14879084. For PathOrderField, use DBSCAN and Sokal-Sneath metric with $minPoints = 5$ and $\epsilon = 0,0375229685$.

DBSCAN	**Hierarchical Clustering**
PathOrderField	PathFieldValue
metric = sokalsneath	metric = dice
$\epsilon = 0,0375229685$	linkage=weighted
minPoints= 5	criterion = 'inconsistent'
	threshold = 1,14879084
	depth = 2

Table 3.6: Algorithms and configurations proposed

Chapter 4

Experiments and Results

4.1 Dataset

The dataset used will be VISION [vis17]. This dataset has been chosen because it was the one used in [ISF+19]. Its main advantage is that it is constituted by smartphone videos and images. It was thought to explore relations between images and videos taken by the same camera, a very interesting problem nowadays with the omnipresence of smartphones. Even if images will be ignored in this work, it is still a good dataset of smartphone videos, and it also includes their transformation to social network.

It is composed by 35 devices, 29 models, and 11 brands. A total of 1914 videos. Of those, only 648 are original, and the rest has been uploaded to a social network and downloaded again. The result are 644 WhatsApp videos, and 622 from Youtube. The result can be seen in Table 4.1. In terms of container analysis, the former videos must be considered as two big classes, WA (WhatsApp) and YT (YouTube). These videos have been reencoded in other containers when uploaded, making the original source untrackable. Also, videos from WhatsApp had been uploaded using an iPhone 7, even if it was not the source. Nevertheless, they still offer a trace of the social network that encoded it.

Brand	Model	Device	# Videos
Apple	iPad2	D12	16
	Ipad mini	D20	16
	iPhone 4	D09	19
	iPhone 4S	D02	13
		D10	15
	iPhone5	D29	19
		D34	32
	iPhone 5C	D05	19
		D14	19
		D18	13
	iPhone 6	D06	17
		D15	18
	iPhone 6 Plus	F19	19
Asus	Zenphone 2 Laser	D23	19
Huawei	Ascend G6-U10	D33	19
	Honor 5C NEM-L51	D30	19
	P8 GRA-L09	D28	19
	P9 EVA-L09	D03	19
	P9 Lite VNS-L31	D16	19
Lenovo	Lenovo P70-A	D07	19
LG	D290	D04	19
Microsoft	Lumia 640 LTE	D17	10
OnePlus	A3000	D25	19
	A3003	D32	19
Samsung	Galaxy S III Mini GT-I8190	D26	16
	Galaxy S III Mini GT-I8190N	D01	22
	Galaxy S3 GT-I9300	D11	19
	Galaxy S4 mini GT-I9195	D31	19
	Galaxy S5 SM-G900F	D27	19
	Galaxy Tab 3 GT-P5210	D08	37
	Galaxy Tab A SM-T555	D35	16
	Galaxy Trend Plus GT-S7580	D22	16
Sony	Xperia Z1 Compact D5503	D12	19
Wiko	Ridge 4G	D21	11
Xiaomi	Redmi Note 3	D24	19
WhatsApp	WhatsApp	WhatsApp	644
Youtube	Youtube	Youtube	622

Table 4.1: Composition of the VISION dataset

4.2 Semimetric Space

The right representation of the data is key part for any data related problem. So, multiple metric (or semimetric) spaces had been considered. A metric or semimetric space is just a set of elements (called points, or videos in our case) and a metric or semimetric that computes a distance between them..

4.2.1 Elements Representation

The proposed representation in [ISF+19] is to work with the combination of PathOrderFieldValue. Nevertheless, once all the formulas have been defined to work with videos as set of elements, the universe of elements composing the videos can be modified. Different representations of the same dataset had been defined. Starting with the reminder of the definition of PathOrderField and PathFieldValue , and then other possible universes will be shown.

- **PathOrderField**
  The structure the atoms, and the contained fields, already offer valuable information. Removing part of the information (the value) weakens the condition for equality, and might allow to ignore smaller diferences. The PathOrderField is defined as the union of the ordered path and the field separated with a '/'. Remember, for example, the first row of Table 3.1 is $ftyp - 1/majorBrand$.

- **PathFieldValue**
  The order of the atoms could not be so relevant, as one missing atom on the root could make two (almost) equals videos completely different. The PathFieldValue is defined as the union of Path and Field separated by a '/' and then the Value separated by a "=" is added. The first row of Table 3.1 is then $ftyp/majorBrand = mp42$.

- **PathOrderFieldValues**
  The most complete element we can get with the information gathered from the file. The PathOrderFieldValue is defined as the union of the PathOrder and Field separated by a '/' and then the Value separated by a "=" is added. The first row of Table 3.1 is then $ftyp - 1/majorBrand = mp42$.

- **PathField**
  From the two previous reasoning, we get the weakest of all the possible elements elements. The PathField is defined as the union of Path and Field separated by a '/'. The first row of Table 3.1 is then $ftyp/majorBrand$.

4.2.2 Metric

One measure is needed for clustering, verifying the conditions below. However, some of the proposed "measures" doees not hold the triangular inequality. These functions are called semimetrics. Clustering in a semimetric space is still possible, at least for the the two chosen methods, so this will not cause probles. Still, semimetrics will be marked.

- $D(x, y) \geq 0$
- $D(x, y) = 0 \Leftrightarrow x = y$
- $D(x, y) = D(y, x)$
- $D(x, z) \leq D(x, y) + D(y, z)$

One extra notation is used additionally to the regular set notation. This notation is very useful when representing set as a binary array where each position corresponds to an element and the 0 or 1 indicates the absence or presence of the element in the set. Then the notation is just product of vectors. This gives place to the following notation:

Given two videos X and Y of a universe Ω with $Z = \bigcup_{X \in \Omega} X$:

$$\begin{aligned} n_{11} =& |S|, \text{ where } S = \{x \mid x \in X, x \in Y\} \\ n_{00} =& |S|, \text{ where } S = \{x \mid x \notin X, x \notin Y, x \in Z\} \\ n_{10} =& |S|, \text{ where } S = \{x \mid x \in X, x \notin Y\} \\ n_{01} =& |S|, \text{ where } S = \{x \mid x \notin X, x \in Y\} \end{aligned} \tag{4.1}$$

- **Dissimilarity semimetric**
  The dissimilarity function defined in [ISF+19] establish a way to compute "distances" between videos. It has been redefined to work with videos as sets, more simple and understandable. As shown in Figure 4.1 it does not hold triangular inequality, even if it holds the rest, so it is a semimetric.

$$\begin{aligned} D\colon X \times Y \rightarrow & \quad [0,1] \\ X \times Y \rightarrow & \quad 1 - \frac{|X \cap Y|(|X| + |Y|)}{2|X||Y|} \\ & = 1 - \frac{n_{11}(2n_{11} + n_{10} + n_{01})}{2((n_{11} + n_{10})(n_{11} + n_{01})} \end{aligned} \tag{4.2}$$

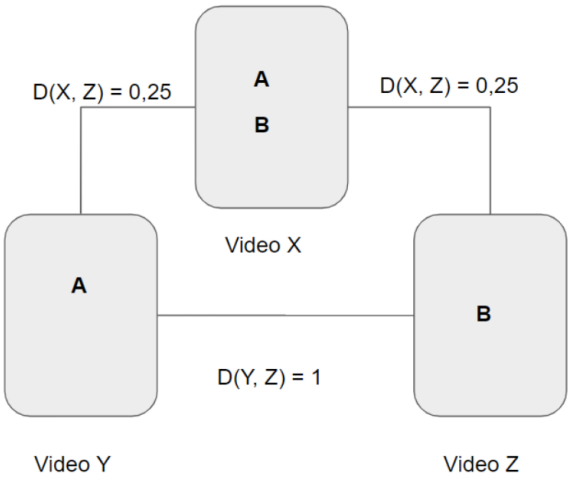


Figure 4.1: Example of schema violating the Triangular Inequality for the Dissimilarity semimetric

- **Jaccard Distamce**
  The Jaccard Distance (JD) is associated to the Jaccard Index (JI). It is a well known metric, and it is defined as:

$$\begin{aligned} JD\colon X \times Y \rightarrow & \quad [0,1] \\ X \times Y \rightarrow & \quad 1 - JI(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \\ & = 1 - \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \end{aligned} \tag{4.3}$$

- **Hamming Distance**
  The Hamming Distance is metric defined as the number of differences between two vectors, and defines a metric. For sets (seen as binary vectors) is the number of elements in X not in Y plus the opposite:

$$\begin{aligned} HD\colon X \times Y \to &\quad \mathbb{N} \\ X \times Y \to &\quad |X| + |Y| - 2|X \cap Y| \\ &= n_{10} + n_{01} \end{aligned} \tag{4.4}$$

- **Cosine Distance**
  The cosine distance (CD) is a semimetric, but not thought for sets. Nevertheless if sets are represented as as binary vectors it is equivalent to the Otsuka-Ochiai coefficient, equivalent to the cosine similarity (CS), and then it gets:

$$\begin{aligned} CD\colon X \times Y \to &\quad [0,1] \\ X \times Y \to &\quad 1 - CS(X,Y) = 1 - \frac{|X \cap Y|}{\sqrt{|X| \times |Y|}} \\ &= 1 - \frac{n_{11}}{\sqrt{(n_{11}+n_{10}) \times (n_{11}+n_{01})}} \end{aligned} \tag{4.5}$$

- **Sørensen–Dice Dissimilarity**
  The Sørensen–Dice dissimilarity (SD) is a semimetric that comes from the Sørensen–Dice coefficient (SDC), and it is defined as:

$$\begin{aligned} SD\colon X \times Y \to &\quad [0,1] \\ X \times Y \to &\quad 1 - SDC(X,Y) = 1 - \frac{2|X \cap Y|}{|X| + |Y|} \\ &= 1 - \frac{2n_{11}}{2n_{11} + n_{10} + n_{01}} \end{aligned} \tag{4.6}$$

- **Sokal-Sneath metric**
  The Sokal-Sneath (SS) metric is defined as:

$$\begin{aligned} SS\colon X \times Y \to &\quad [0,1] \\ X \times Y \to &\quad \frac{2(|X \cup Y| - |X \cap Y|)}{2|X \cup Y| - |X \cap Y|} \\ &= \frac{2(n_{11} + n_{10} + n_{01})}{3n_{11} + 2n_{10} + n_{01}} \end{aligned} \tag{4.7}$$

- **Euclidean metric**
  The classic euclidean metric (L2) applied to sets viewed as binary vectors:

$$\begin{aligned} L2\colon X \times Y \to &\quad \mathbb{R}^+ \\ X \times Y \to &\quad \sqrt{|X| + |Y| - 2|X \cap Y|} \\ &= \sqrt{n_{10} + n_{01}} \end{aligned} \tag{4.8}$$

- **Yule Dissimilarity**
  The Yule Dissimilarity (YD) is a semimetric that cannot be explained only in terms of X and Y alone, it also depends on the number of unique elements contained in the union of all the videos of the universe, $N = |\bigcup_{Z \in \Omega} Z|$. So this metric is dependent on the the universe of videos, taking in account not only what they do have in common, but what they both miss.

$$\begin{aligned} YD\colon X \times Y \to & \quad \mathbb{R}^+ \\ X \times Y \to & \quad \frac{2(|X| - |X \cap Y|)(|Y| - |X \cap Y|)}{|X \cap Y|(N - |X| - |Y|) + |X||Y|} \\ & = \frac{2n_{10}n_{01}}{(n_{10}n_{01})(n_{11}n_{00})} \end{aligned} \tag{4.9}$$

Other alternatives have been thought, as almost any metric, semimetric, o function can be applied, as minkowski distances, canberra, etc. Many were tested too: Manhattan, Canberra, Chebyshev, Correlation, Kulsinski, Rogers-Tanimoto, Russell-Rao, and Sokal-Michener but they revealed no interest.

Of those some are equivalent when applied to sets, as Canberra and Manhattan (equivalent to Hamming), or Chebyshev, that will be equivalent to discrete metric. Even any ρ-minkowski metric, will only be equivalent to the euclidean one.

4.2.3 Filtering

Apart from the variations of the initial elements, it is worth to consider filtering some by various reasons. Some atoms might be kept apart because they are vulnerable to modifications. The removed atoms are: *mdta*, *udta*.

Also, if the Value is part of the element, it could not be representative to consider fields related to modification time, size of the file, etc. Those fields are different in every video because they represent unique characteristics that will never be repeated. It is just noise. However, even if the the Value not is part of the element, it could be interesting to remove some of these fields, as it has been seen.

The removed fields are: *uuid*, *payload*, *notEspecified*, *xyz*, *entries*, *size*, *modificationTime*, *stuff*, *creationTime*, *duration*, *entryCount*, *sampleCount*, *unkn*, *freeSpace*.

4.2.4 Oversampling

As stated before, the data set is composed by 35 devices, 29 models and 11 brands. Plus the two huge classes of 638 videos from YT and WA. Even ignoring those two, the number of samples per class (at any level) is not balanced as shown in Table 4.1.

This can cause some classes to bias the result, by having a greater impact on the result just because their size.

To deal with, data set is oversampled. For oversampling, elements from minority classes will be duplicated until the class matches the size of the biggest one. Nevertheless, this is not always possible due to memory limitations. The resulting size of the data set is too big. For example, considering videos by model, oversamplig produces 19964 samples (ten times more than initially).

4.3 Selection of the Representation of the Data

In this section the experimental method to select the appropriate (semi)metric space is explained. It has also been take advantage of to choose label work with, and to filter or not.

4.3.1 Evaluation for Elements and Metric Selection

In order to choose the best representation of the dataset to cluster, and the best metric, the alternatives had be tested with the silhouette coefficient. The representation and measure with the highest silhouette coefficient will be the most likely to be correctly separated.

The silhouette coefficient is a measure of the consistency of the clusters. It measures both the cohesion and separation of the clusters. It is computed for every point and then the average is taken.

For every point in the data set $i \in C_k$.

$$a(i) = \frac{1}{|C_k| - 1} \sum_{j \in C_k, j \neq i} d(i,j) \tag{4.10}$$

$$b(i) = min_{l \neq k} \frac{1}{|C_l|} \sum_{j \in C_l} d(i,j) \tag{4.11}$$

$$s(i) = \begin{cases} 1 - a(i)/b(i) & a(i) < b(i) \\ 0 & a(i) = b(i) \\ a(i)/b(i) - 1 & a(i) > b(i) \end{cases} \tag{4.12}$$

4.3.2 Configuration of the Experiment

All the posible metric and semimetric spaces had been putted to the test, as shown in left side of Table 4.2. Additionally, each metric space will be tested for each possible label, to see the viability of grouping videos at different levels, and with and without filtering, as shown in the right side. This gives a total of 192 executions.

For each execution, silhouette coefficient of all the videos in the dataset and just the original ones were computed, to see which is the most suitable representation, level of grouping and preprocessing to work with.

On the experiment performed only with original videos those were oversampled to extend the minority classes to match the size of the biggest one.

Elements	**Metric**		**Label**	**Filtering**
PathField	Dissimilarity		Brand	Yes
PathOrderField	Jaccard		Model	No
PathFieldValue	Hamming		Device	
PathOrderFieldValue	Cosine			
	Dice			
	Sokal-Sneath			
	Yule			
	Euclidean			

Table 4.2: Table with the possible elements and metrics. Added to them, the possible labels of each observation of the generated semimetric space, and the possible filtering

4.3.3 Results

The table with the whole results can be found in Table A.1. The maximum (for any metric) silhouette coefficient found for each combination can be found in Table 4.3, alongside with the filter. It can be seen that the values with the full dataset and only with original videos do not always go along.

Therefore, in Table 4.4 the maximum (for any metric) silhouette coefficient of the average between the full dataset and only the original videos are represented. One configuration over stands, the PathOrderField at brand level with filtering, and other 3 that are close, as shown in Table 4.5. PathOrderFieldValue, even if used at [ISF+19] has an ordinary silhouette coefficient, so it will be left apart.

Nonetheless, it is important to see consistent results for the different metrics and filtering. Important variations are expected, but outliers might point unexpected behaviours. In Table 4.6 the top 3 metric for each of the most promising configurations shown in 4.5. It can be seen that the silhouette coefficient is slightly affected by the chosen metric, with the exception of Yule. Remember that Yule semimetric has the particularity of using the n_{00} from equation 4.1.

		Filter			
		No		**Yes**	
Dataset	**Class**	**Full dataset**	**Original only**	**Full dataset**	**Original only**
PathField	Brand	0,726358	0,045549	0,250225	0,398712
PathField	Device	0,668669	0,22359	-0,044316	0,280359
PathField	Model	0,622528	0,646994	-0,104571	0,086335
PathFieldValue	Brand	0,653203	0,559167	0,661963	0,686108
PathFieldValue	Device	0,655339	0,449343	0,696647	0,417425
PathFieldValue	Model	0,615433	0,555977	0,647963	0,589758
PathOrderField	Brand	0,765278	0,239737	0,747372	0,67382
PathOrderField	Device	0,686581	0,239942	0,666356	0,259924
PathOrderField	Model	0,630161	0,665327	0,609536	0,226282
PathOrderFieldValue	Brand	0,536545	0,414133	0,588605	0,625443
PathOrderFieldValue	Device	0,473966	0,385987	0,539669	0,300273
PathOrderFieldValue	Model	0,432278	0,495649	0,488659	0,517986

Table 4.3: Table of the maximum silhouette coefficient per combination for any metric with the top 3 values for the full dataset and only with original videos highlighted

		Filter	
Dataset	**Class**	**No**	**Yes**
PathField	Brand	0,3859535	0,220807
PathField	Device	0,445123	0,1180215
PathField	Model	0,634761	-0,009118
PathFieldValue	Brand	0,606185	0,6740355
PathFieldValue	Device	0,552341	0,557036
PathFieldValue	Model	0,585705	0,6188605
PathOrderField	Brand	0,4989525	0,710596
PathOrderField	Device	0,4632615	0,46314
PathOrderField	Model	0,647744	0,417909
PathOrderFieldValue	Brand	0,453314	0,60656
PathOrderFieldValue	Device	0,383566	0,395454
PathOrderFieldValue	Model	0,462516	0,4527485

Table 4.4: Table of the maximum average silhouette coefficient per combination for any metric. Top 4 values are highlighted

Elements	**Label**	**Filter**	**Silhouette**
PathOrderField	Brand	Yes	0,711
PathFieldValue	Brand	Yes	0,674
PathOrderField	Model	No	0,648
PathField	Model	No	0,635

Table 4.5: Most promising configurations based on the maximum average between both executions. The three most promising results for the full dataset and only for original videos are highlighted in green and yellow

Elements	**Label**	**Filter**	**Metric**	**Full**	**Original**	**Avg**
PathOrderField	Brand	Yes	euclidean	0,747	0,673	0,710
PathOrderField	Brand	Yes	sokalsneath	0,732	0,661	0,697
PathOrderField	Brand	Yes	jaccard	0,730	0,647	0,689
PathOrderField	Model	No	euclidean	0,630	0,665	0,647
PathOrderField	Model	No	sokalsneath	0,618	0,655	0,636
PathOrderField	Model	No	jaccard	0,611	0,641	0,626
PathFieldValue	Brand	Yes	yule	0,661	0,686	0,674
PathFieldValue	Brand	Yes	dice	0,490	0,559	0,525
PathFieldValue	Brand	Yes	cosine	0,490	0,558	0,524
PathField	Model	No	euclidean	0,622	0,646	0,634
PathField	Model	No	sokalsneath	0,609	0,642	0,625
PathField	Model	No	dissimilarity	0,605	0,642	0,624

Table 4.6: Top 3 metrics of the most promising configurations based on the maximum average between both executions

4.3.4 Conclusion

The most promising configurations had been exposed on Table 4.6 and for the next experiments limit to them. In Figure 4.2 the result of the silhouette analysis for PathOrderField and Sokal-Sneath metric are shown for original videos is shown as an example. It can be seen that except for Huawei and Xiaomi, the classes have positive silhouette coefficients, even perfect ones.

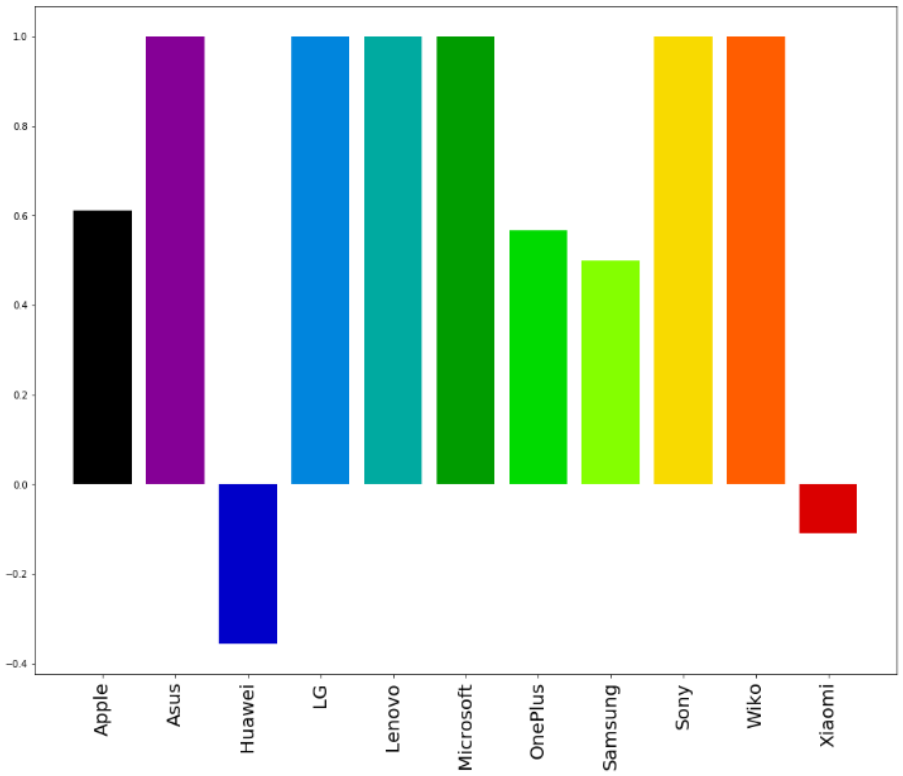


Figure 4.2: Silhouette coefficient for all brands in the dataset as PathOrderField with metric Sokal-Sneath and filtering

4.4 Evaluation for Clustering Performance

The question of how to measure the performance of the clustering is important in order to compare the multiple algorithms and configurations. To do so, a function to compare the predicted clusters with the real classes of the videos is needed. Only informed measures had been used, depending on the real labels of the elements, because uninformed techniques were empirically useless because of the dispersion of the data.

The desired properties of the measure include [VEB10]:

- Normalisation
  As different metrics and representations of the dataset are used, this is a mandatory property to be valid for comparison.

- Constant baseline
  A measure that is biased by "luck" will not be useful. Many of the common measures tend to higher values when the size of the clusters decreases (N/K is low).

Other properties as the measure to define a metric, can be a nice to have, but will not impact on our results.
Clusters are no more than partitions over the universe X of videos (with $|X| = N$). That is: $\mathbf{U} = \{U_i\}_{i \in K}$ such as: $\bigcup_{U \in \mathbf{U}} U = X$ and $\forall A, B \in \mathbf{U}, A \neq B \implies A \cap B = \emptyset$.

Over all the possible approaches, two over stand:

- Pair counting
  For every pair of points in the universe, $\binom{N}{2}$, they can belong to the same cluster in both partitions $\mathbf{U}$ and $\mathbf{V}$ (N_{11}), be in different clusters in both partitions (N_{00}) or be in the same cluster in one partition and not in the other (N_{01} or N_{10}).
- Information Theory
  Using the entropy, conditional entropy and joint entropy, mutual information offers the measure of shared information between the partitions.

The Adjusted Mutual Information (AMI) has been selected because it works better for unbalanced cluster size [RVBV16]. The V-Measure has been monitored too as it offers an insight on what is going on.

4.4.1 Mutual Information (MI)

The mutual information is a measure of the shared information between two partitions. It relies on information theory and entropy calculus to do so.

The comparison of two partitions can be summarised in a contingency table as shown in the example in Figure 4.3. The partitions are $\mathbf{U}$ and $\mathbf{V}$ over X with $|\mathbf{U}| = R, |\mathbf{V}| = C$. Then:

$$\forall i \in [[1, R]], |U_i| = a_i \tag{4.13}$$
$$\forall j \in [[1, C]], |V_j| = b_j \tag{4.14}$$
$$\forall i, j \in [[1, R]] \times [[1, C]], |U_i \cap V_j| = n_{ij} \tag{4.15}$$

Sex \ Handedness	Right handed	Left handed	Total
Male	43	9	52
Female	44	4	48
Total	87	13	100

(a) Simple example

$\mathbf{U} \backslash \mathbf{V}$	V_1	V_2	…	V_C	Sums
U_1	n_{11}	n_{12}	…	n_{1C}	a_1
U_2	n_{21}	n_{22}	…	n_{2C}	a_2
⋮	⋮	⋮	⋱	⋮	⋮
U_R	n_{R1}	n_{R2}	…	n_{RC}	a_R
Sums	b_1	b_2	…	b_C	$\sum_{ij} n_{ij} = N$

(b) Notation

Figure 4.3: Contingency table

For every random element of X, the probability that the object falls into cluster U_i is:

$$P_{\mathbf{U}}(i) = \frac{|U_i|}{N} = \frac{a_i}{N} \tag{4.16}$$

Thus, the entropy associated with the partitioning $\mathbf{U}$ is:

$$H(\mathbf{U}) = -\sum_{i=1}^{R} P_{\mathbf{U}}(i) \log P_{\mathbf{U}}(i) \tag{4.17}$$

$H(U)$ is non-negative and takes the value 0 only when there is no uncertainty determining an object's cluster membership, that, when there is only one cluster. Similarly, the entropy of the clustering $\mathbf{V}$ can be calculated as:

$$H(\mathbf{V}) = -\sum_{j=1}^{C} P_{\mathbf{V}}(j) \log P_{\mathbf{V}}(j) \tag{4.18}$$

where $P_{\mathbf{V}}(j) = \frac{b_j}{N}$.

The mutual information between the two partitions:

$$MI(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{R} \sum_{j=1}^{C} P_{\mathbf{U}\cap\mathbf{V}}(i,j) \log \frac{P_{\mathbf{U}\cap\mathbf{V}}(i,j)}{P_{\mathbf{U}}(i) P_{\mathbf{V}}(j)} \tag{4.19}$$

where $P_{\mathbf{U}\cap\mathbf{V}}(i,j) = \frac{|U_i \cap V_j|}{N} = \frac{n_{ij}}{N}$

MI is a non-negative quantity, quantifying the shared information between the two partitions. Nevertheless, it does not hold the normalisation property, but its upper bounded by $H(\mathbf{U})$ and $H(\mathbf{V})$.

A normalised version of the Mutual Information exists, called Normalised Mutual Information (NMI):

$$NMI = \frac{MI(\mathbf{U}, \mathbf{V})}{max\{H(\mathbf{U}), H(\mathbf{V})\}} \tag{4.20}$$

Then, the result lies in the range $[0, 1]$. Besides the improvement, it still fails the "constant baseline" property. As the ratio N/K decreases, the Rand Index tends to increase even for random clusters.

4.4.2 Adjusted Mutual Information (AMI)

As stated before, the Mutual Information tends to lie on higher values under certain constraints. To deal with, the Adjusted Mutual Information arises. It takes into account the expected value $E\{MI(U,V)\}$ for a random partition to correct the MI.

$$\begin{aligned} E\{MI(U,V)\} = & \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N} \log\left(\frac{N \cdot n_{ij}}{a_i b_j}\right) \times \\ & \frac{a_i! b_j! (N-a_i)! (N-b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!} \end{aligned} \tag{4.21}$$

where $(a_i + b_j - N)^+$

The resulting Adjusted Mutual Information does hold both the normalisation and the constant baseline.

$$\overbrace{AMI(U,V)}^{\text{Adjusted MI}} = \frac{\overbrace{MI(U,V)}^{\text{MI}} - \overbrace{E\{MI(U,V)\}}^{\text{Expected MI}}}{\underbrace{\max\{H(U), H(V)\}}_{\text{MI upeerbound}} - \underbrace{E\{MI(U,V)\}}_{\text{Expected MI}}} \tag{4.22}$$

4.4.3 V-Measure (VM)

This is the harmonic mean of the homogeneity and completeness of the predicted clusters with respect to the true labels. It can be seen as a kind of F-Score for clustering evaluation. It is based on the MI and the entropy of each partition. Lets say **U** are the predicted labels, and **V** the true ones.

$$homogeneity = \frac{MI}{H(\mathbf{V})} \tag{4.23}$$

$$completeness = \frac{MI}{H(\mathbf{U})} \tag{4.24}$$

$$VM = \frac{2(homogeneity \times completne^{o}ss)}{(homogeneity + completeness} \tag{4.25}$$

The V-measure does hold the normalisation property, but not the constant baseline, as the MI. This measure is, even so, interesting as an insight on the results.

4.5 DBSCAN

DBSCAN algorithm presented in [EKSX96], and introduced in chapter 3, had been used to try to cluster video files.

4.5.1 Configuration of the Experiment

The 12 configurations shown in Table 4.6 had been used to run DBSCAN. The remaining parameters are epsilon value, ϵ, and $minPoints$. The latter will be fixed to 5, as DBSCAN main difference with Hierarchical Clustering is its ability to ignore noise. Epsilon, however, has been varied.

The range of variation of ϵ depends on the metric: for metrics taking values in range $[0, 1]$, values from 10^{-5} to 1 had been tested, for euclidean values had gone from 0.1 to $\sqrt{|\bigcup_{X \in \Omega} X|}/10$, one tenth of the maximum possible value. Notice that the minimum value of epsilon for euclidean is equivalent to 0, as the minimum distance between two non equal binary vectors is ≥ 1. Zero was not directly been tested because it was technically not allowed. Along this range 250 values were tested, distributed in a logarithmic scale.

Two different executions had been performed, using different subsets for train and test. Even if clustering is not supposed to have a training, the need to look for the adequate value for ϵ, and the use of informed measures (as AMI) forces us to do so. The multiple sets combination are shown in Table 4.7.

4.5.2 Results

The complete results can be found in appendix B.

Train	Test	Results
66% Full	33% Full	Table B.1
66% Original (oversampled)	33% Original (oversampled)	Table B.2

Table 4.7: Different combinations of train and test subsets and location of table with results

It is interesting to notice that the maximum AMI values for each configuration is pretty independent of the metric. The only exception is the case of the Yule dissimilarity. Yule has the particularity of looking at the the elements absent on both videos, so it is expected to have different behaviours. Its results, however, are quite deceiving.

At Model level configurations, it is also interesting that the max AMI value is achieved at very low epsilon values. It is in fact equivalent to $\epsilon = 0$, and it will have the same behaviour with any metric. The resulting clusters made can be seen in Figure 4.4.

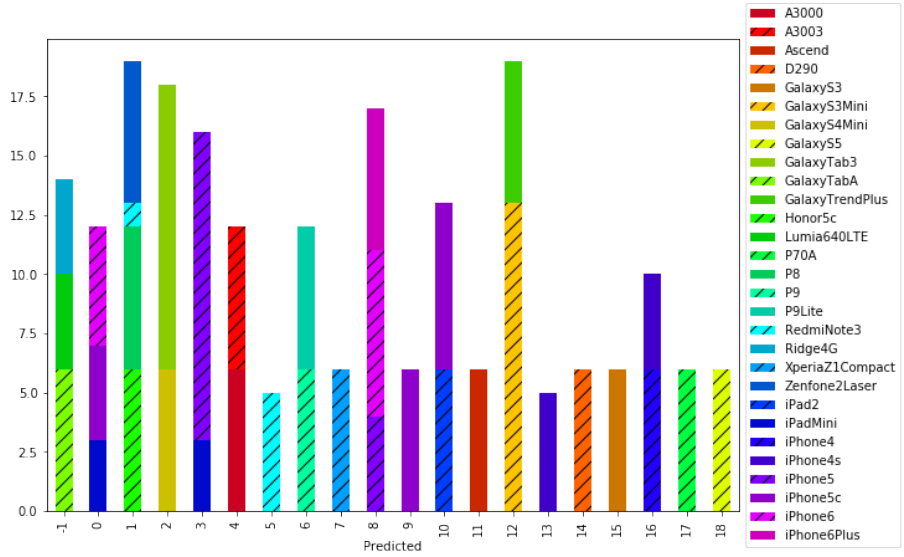


Figure 4.4: Result of the clustering with DBSCAN of the 33% of the original part of the dataset as PathOrderField with euclidean metric and $\epsilon = 0$ equivalent to discrete metric

At Brand level, the best performance is different for the original dataset and for the full dataset. This is because the original is oversampled, and the full not, so there is a bias with Apple, Youtube and WhatsApp as majority classes. For the full dataset, the highest value is with PathFieldValue, dice metric, and $\epsilon = 0,1038$. However in Figure 4.7 it can be seen that the clusters are just right for the majority classes.

Nevertheless, the top AMI value for the original dataset is with PathOrderField, Sokal-Sneath metric and $\epsilon = 0,0375$, producing the clusters shown in Figure 4.5. Figure 4.6 shows the result of this clustering for the test subset of the full dataset. It can be seen that it correctly correctly groups Whatsapp and Youtube videos, even if they where not in the training set that produced it. The AMI of this configuration evaluated with the 33% of the full dataset is 0,80, not so far from the 0,81 achieved by the previous one.

Regarding the V-Measure, we can see that this configuration has an homogeneity of 0,893 and a completeness of 0,688, indicating that the we can "trust" that two grouped together videos belong to the same brand, but not so much the opposite.

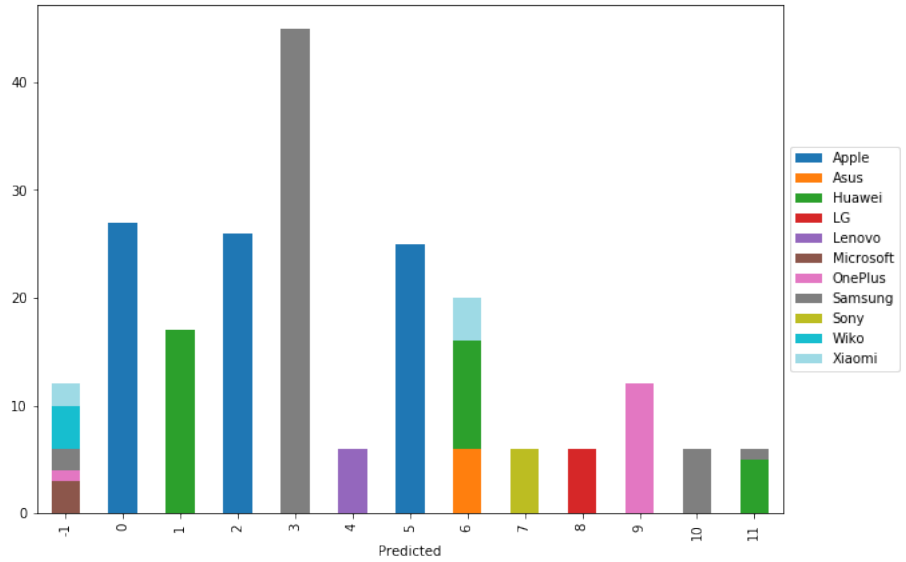


Figure 4.5: Result of the clustering with DBSCAN the 33% used for testing of the original part of the dataset as PathOrderField with Sokal-Sneath metric and $\epsilon = 0,0375$

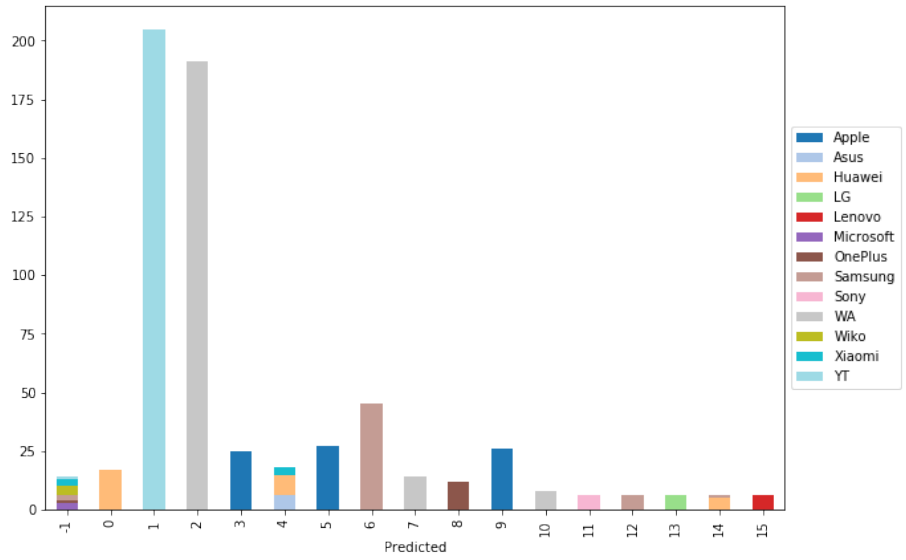


Figure 4.6: Result of the clustering with DBSCAN the 33% used for testing of the full dataset as PathOrderField with Sokal-Sneath metric and $\epsilon = 0,0375$

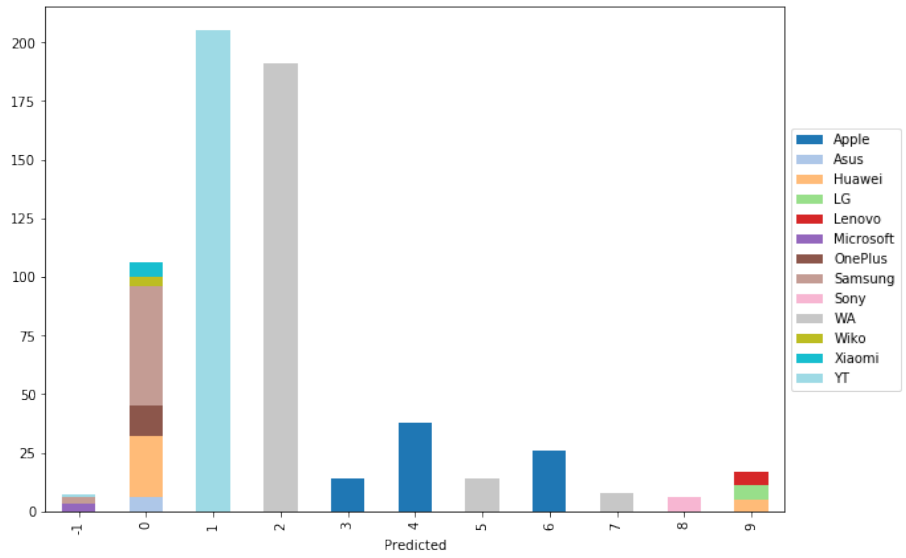


Figure 4.7: Result of the clustering with DBSCAN the 33% used for testing of the full part of the dataset as PathFieldValue with dice metric and $\epsilon = 0,1038$

4.5.3 Conclusion

The results obtained at Model level (Figure 4.4) show the best we can achieve at this level with PathOrderField or PathField, as the videos grouped together are identical. Grouping those videos by model seems then impossible as they lack inter-class variability. Further details will be seen in next section.

At Brand level, however, results were more encouraging. The configuration shown in 4.8 achieved to separate correctly Whatsapp, Youtube, LG, Lenovo and Sony videos, even if lacking completeness. Some brands were more or less well separated, as Huawei and Samsung, and others where directly classified as noise, namely Wiko and Microsoft. The performance of the configuration is shown in 4.9.

Algorithm	**DBSCAN**
Representation	PathOrderField
Filtering	Yes
Metric	Sokal-Sneath
eps	0,037522968
minPoints	5

Table 4.8: Configuration of the DBSCAN experiment that performed the best

	33% Original	**33% Full**
Classes P / R	13 / 11	17/13
AMI	0,648	0,780
V-Measure	0,777	0,876
Homogenity	0,893	0,959
Completness	0,688	0,807

Table 4.9: Results of the execution of the proposed clustering algorithm shown in Table 4.8

4.6 Hierarchical Clustering

Hierarchical Clustering introduced in chapter 3, had been used to try to group video files. This method allows also to draw the hierarchy of linkages between video files, so it offers more valuable information.

4.6.1 Configuration of the Experiment

The 12 configurations shown in Table 4.6 had been used to run Hierarchical Clustering. The remaining parameters are the linkage criterion, the depth d to compute the inconsistency and the threshold t to cut. Multiple linkage criteria will be tested, and the threshold t will vary too. The depth, however, will be fixed to $d = 2$. Higher values are discouraged since they can be affected by the number of observations.

The threshold t will vary from 10^{-1} to 10^{1}, and the linkages methods are those described in 3.2. This threshold measures, as explained in chapter 3, how the h "jumps" of the dendrogram vary, what are intuitively seen as unusual "big" jumps. Along this range 250 t values were tested, distributed in a logarithmic scale.

Two different executions had been performed, using different subsets for train and test. Even if clustering is not supposed to have a training, the need to look for the adequate value for ϵ using informed metrics (as AMI) forces us to do so. The multiple sets combination are shown in Table 4.10.

4.6.2 Results

The full results are in appendix C, as summarised in Table 4.10.

Train	Test	Results
66% Full	33% Full	Table 1
66% Original (oversampled)	33% Original (oversampled)	Table 2

Table 4.10: Different combinations of train and test subsets and location of table with results

As for DBSCAN at Model level, the max AMI value is at the lowest threshold possible. It is still equivalent to cut the dendrogram at 0, so the metric and linkage are irrelevant. In fact, the only difference is that DBSCAN with consider the clusters smaller that 5 elements as noise, because of the *minPoints* parameter.

For PathOrderField with filter, using Sokal-Sneath and a very low threshold (0,1) and single linkage results are shown in Figure 4.8. Thanks to the linkage performed as part of the hierarchical clustering, the dendrogram shown in Figure 4.9 allows to see some details. The clusters on the left (clusters 1 to 7), correspond to videos from Apple models, are pretty different of other brands, as they merge together at height 0,6, but with other brands only at almost heigth 1. The OnePlus brand is also distinguishable, as the Sony Xperia or the Wiko Ridge4G. Other devices from different brands, on the contrary, show no difference with this representation, as the Asus Zenfone, producing videos exactly like the ones from Huawei's Honor 5c and P8. It is always impossible to distinguish an Asus video from a Huawei one with this representation of the data.

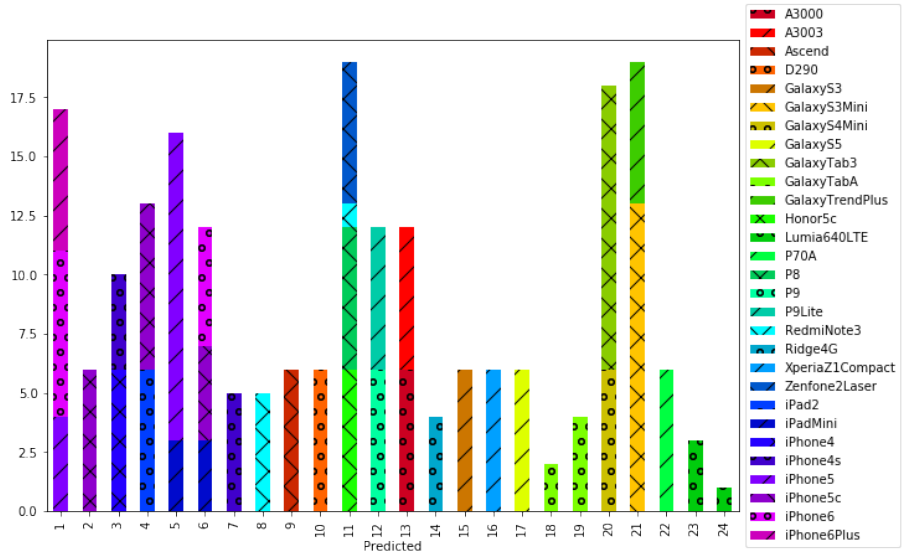


Figure 4.8: Result of the clustering with Hierarchical Clustering of the 33% used for testing of the original part of the dataset as PathOrderField unfiltered with Sokal-Sneath metric and $threshold = 0.1$ and linkage single equivalent to discrete metric

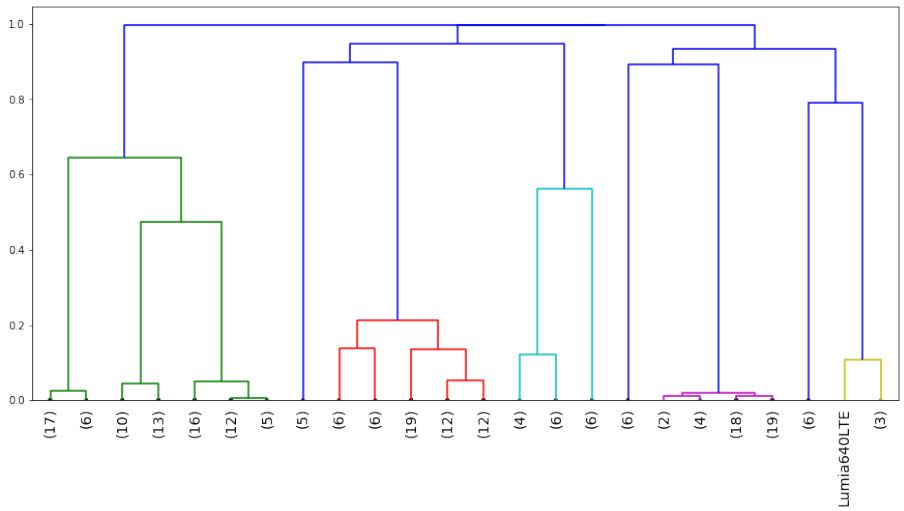


Figure 4.9: Dendrogram linking the resulting clusters from the clustering with Hierarchical Clustering of the 33% used for testing of the original part of the dataset as PathOrderField unfiltered with Sokal-Sneath metric and $threshold = 0.1$ and linkage single equivalent to discrete metric

At Brand level, the highest AMI for both the full dataset and the original only came, again with the threshold at the lowest level, $threshold = 0,1$ for each of the suggested metrics. The results look alike the one in Figure 4.8, but not exactly the same, as the filter was applied to the data. Results are shown in Figure 4.10 for original videos only and for the full dataset in Figure 4.12. The cluster produced are not to complete, but very homogeneous, as only Asus, Huawei and Xiaomi are messed together (and one Samsung video grouped with many Huawei's).

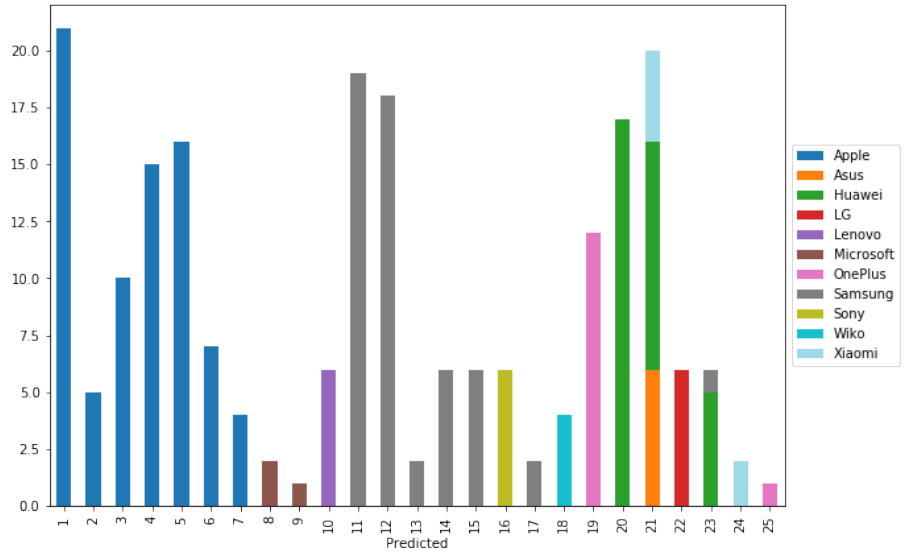


Figure 4.10: Result of the clustering with Hierarchical Clustering of the 33% used for testing of the original part of the dataset as PathOrderField filtered with Sokal-Sneath metric and $threshold = 0.1$ and linkage single equivalent to discrete metric

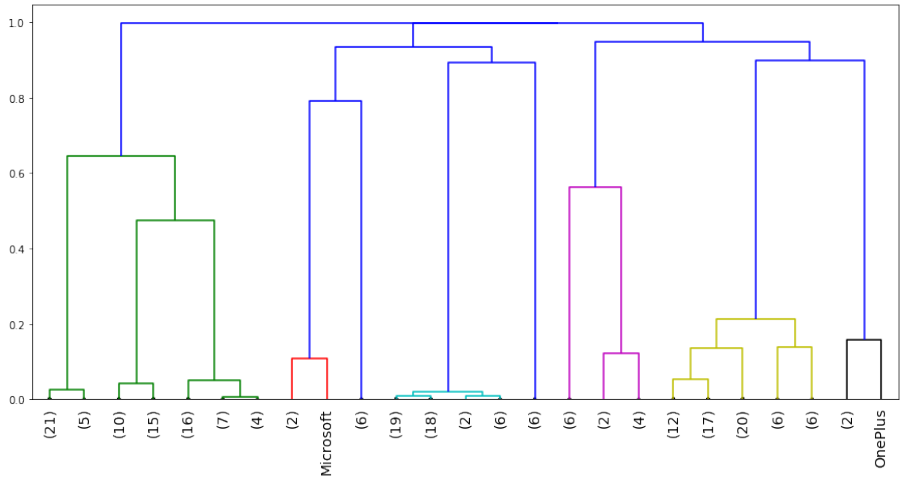


Figure 4.11: Dendrogram linking the resulting clusters from the clustering with Hierarchical Clustering of the 33% used for testing of the original part of the dataset as PathOrderField filtered with Sokal-Sneath metric and $threshold = 0.1$ and linkage single equivalent to discrete metric

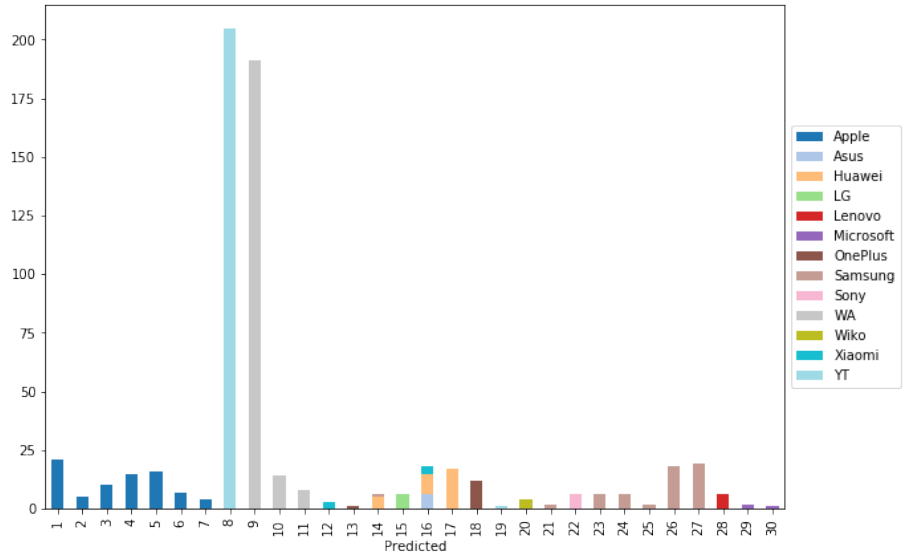


Figure 4.12: Result of the clustering with Hierarchical Clustering of the 33% used for testing with the full dataset as PathOrderField filtered with Sokal-Sneath metric and $threshold = 0.1$ and linkage single equivalent to discrete metric

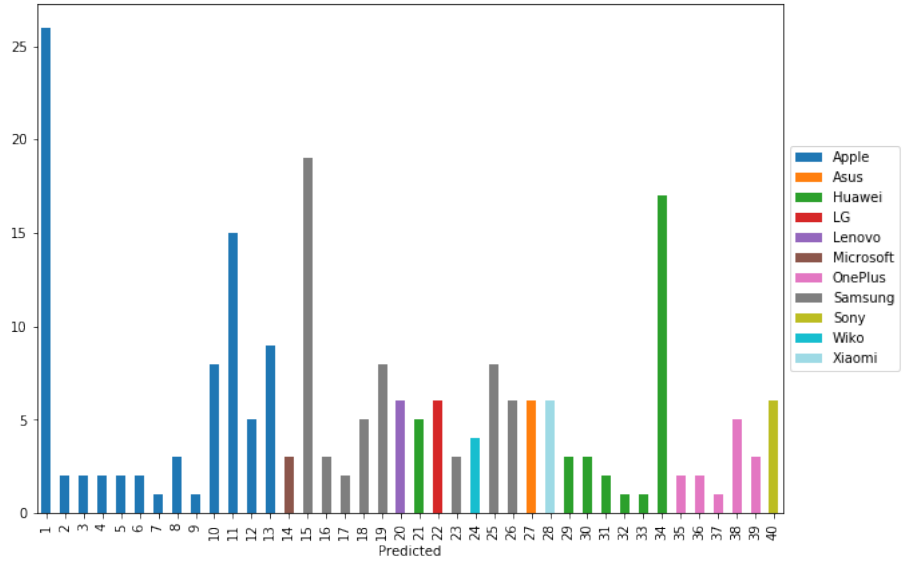


Figure 4.13: Result of the clustering with Hierarchical Clustering of the 33% of the full dataset as PathFieldValue filtered with Dice metric and $threshold = 0.1$ and linkage weighted

An interesting result is, nevertheless, offered by the PathFieldValue representation of the dataset. Using weighted linkage and dice metric, with $threshold = 1,1490$ for inconsistency, the resulting grouping is completely homogeneous. There are 40 clusters for the original dataset, with only 11 brands and 214 videos, but it is still an interesting result as shown in 4.13. When evaluated with videos from Youtube and Whatsapp it is still homogeneous, but the number of classes increases to 149, with only 632 observations, around 4 videos per cluster.

4.6.3 Conclusions

Thanks to the usage of the inconsistency as a way to cut the dendrogram it was achieved to get an homogeneous clustering method, that can ensure that two videos were recorded with a device of the same brand if grouped together. This configuration, shown in Table 4.11, however, is not as good as it seems because of the poor completeness of the result. The measure of the performance of this method can be found in Table 4.12. It is still a good indicator of how the amount of information in PathFieldValue (with filter) is enough to distinguish between video files source.

Algorithm	**Hierarchical Clustering**
Representation	PathFieldValue
Metric	Sørensen-Dice
Filtering	Yes
Linkage	Weighted
Threshold	1,14879084
Criterion	Inconsistent
Depth	2

Table 4.11: Configuration of the experiment that showed a perfect homogeneity

	33% Original	**33% Full**
Classes P / R	40 / 11	152 / 13
AMI	0,442	0,263
V-Measure	0,704	0,539
Homogenity	1,000	1,000
Completness	0,543	0,369

Table 4.12: Results of the execution of the proposed clustering algorithm shown in Table 4.11

Chapter 5

Conclusions and Future Work

This chapter summarise the findings and achievements of this piece, and establish lines of actions for further works.

5.1 Conclusions

Along this document it has been shown how video file information can be exploited to group videos by data source, without previous training of a classifier. The most important part was proven to be the correct processing of the data before clustering. As shown, the correct prepossessing of the data grant the adequate separation needed for the clustering. This same criteria is also applicable to any future technique that attempts to draw conclusions using video container information.

The usage of simple algorithms was also proven effective separating video files by brand. The results were positive, and an algorithm was proven to be able to correctly group videos in homogeneous clusters per brand, even if too many clusters appeared. Another algorithm was also presenting offering a way better classification in general terms even if not having the interesting property of homogeneity.

5.2 Future Work

On the same direction established by this work, some development lines arise.

- **Combine elements**
  The representations of the videos seen can potentially be combined. Looking for alternative representations of videos that combine strengths can be useful. For example, combine PathOrderField and PathOrderFieldValue using the former for fields that are expected to have different values. Or even developing metrics that combine multiple representations.

- **Increase the dataset**
  The current dataset is very limited. The strength of the unsupervised clustering is in the ability to deal with unseen elements. Augmenting the size of the dataset, both on number of videos per device and devices itself can be useful.

- **Cross-validation for unseen devices**
  If the dataset is enlarged, cross validation can be used to increase reliability by completely removing one of the classes at each time. It is important to completely remove the device, to simulate the device too be completely new.

- **Classification or Integrity Validation**
  The defined representations of videos can be adopted to other problems. For example, classification, or integrity validation. There is no reason why the approaches to video representation can not be valid for it.

Chapter 6

Introduccion

6.1 Motivación

A medida que aumenta el numero de dispositivos móviles, y la mayoría de ellos equipados con una cámara, se vuelve más fácil generar contenido de video, y se ha demostrado que es una forma muy efectiva de comunicarse [Bec15]. Las previsiones lo presentan como una tendencia creciente [Cis18]. Ya representa el 75% del tráfico web, y se cree que este número aumentará a 82% para 2021. Según el ranking de Alexa [Ale], YouTube es el segundo sitio web por tráfico en diciembre de 2018, agregando más de 300 horas de video por minuto [Asl18]. Además, en el top 20 de sitios web por tráfico podemos encontrar siete redes sociales que incluyen videos (Facebook, Qq, Twitter, Reddit, Vk, Weibo e Instagram). Además, la videovigilancia es ahora una tendencia, no solo en el hogar o las oficinas, sino también para fortalecer la seguridad pública. Este es el caso de Londres, con más de 500 000 cámaras instaladas, o el futuro de Chine con el programa "Xue Liang" ("Sharp Eyes").

La importancia del análisis forense de videos está ligada a este fenómeno, ya que también aumenta el número de evidencias de video en asuntos legales. Sin embargo, debido a las mejoras en las tecnologías informáticas y la red, se pueden manipular y compartir datos de video tan fácilmente que "los videos digitales y las fotografías ya no pueden considerarse una prueba de evidencia, ya que no se puede confiar en su origen e integridad" [MFB+12]. Se necesitan herramientas para autenticar el contenido de video, trazar el ciclo de vida, identificar la fuente, etc.

Los contenedores de video poseen información relevante sobre la fuente del archivo. Es importante no confundir la información del contenedor de video con los metadatos, ya que el primero utiliza información específica a prueba de manipulaciones de la estructura del archivo de video. La mayor parte del trabajo anterior se centra en el análisis del contenido del video en lugar del contenedor, incluso si este análisis es mucho más pesado.

6.2 Contexto

El presente Trabajo Fin de Grado se enmarca dentro de un proyecto de investigación titulado RAMSES aprobado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 (Convocatoria H2020-FCT-2015, Acción de Innovación, Número de Propuesta: 700326) y en el que participa el Grupo GASS del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad

y Sistemas, http://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM).

Además de la Universidad Complutense de Madrid participan las siguientes entidades:

- Treelogic Telemática y Lógica Racional para la Empresa Europea SL (España)
- Ministério da Justiça (Portugal)
- University of Kent (Reino Unido)
- Centro Ricerche e Studi su Sicurezza e Criminalità (Italia)
- Fachhochschule fur Offentliche Verwaltung und Rechtspflege in Bayern (Alemania)
- Trilateral Research & Consulting LLP (Reino Unido)
- Politecnico di Milano (Italia)
- Service Public Federal Interieur (Bélgica)
- Universität des Saarlandes (Alemania)
- Dirección General de Policía - Ministerio del Interior (España)

6.3 Objetivos

En este trabajo propondremos dos algoritmos capaces separar videos por fuente de codificación, concretamente por marca del dispositivo. Para ello, se siguieron algunos pasos.

- Investigaciónr sobre y formatos de archivo de video
- Diseño de una solución
- Evaluación de la representación del conjunto de datos
- Evaluación de algoritmos de clustering
- Interpretación de los resultados

6.4 Trabajos Relacionados

No hay mucho trabajo previo sobre el análisis de contenedores de video, ya que no se introdujo hasta 2014 en [GFK14]. Este documento presenta la información contenida en los formatos de archivos de video MP4, MOV y AVI. No tiene un enfoque matemático y se limita a exponer las diferencias entre dispositivos, y da cuenta de su potencial para fines forenses de video.

Este primer trabajo, aún en etapas muy tempranas, fue desarrollado por [SPC17] que resultó en el artículo [ISF+19], con el objetivo de verificar la integridad del video y la identificación y clasificación de la fuente. Este último artículo fue la referencia de este trabajo, ya que es la producción más desarrollada sobre este tema.

Incluso si el contenedor de video nunca se había utilizado con fines forenses, existe una gran cantidad de trabajo sobre el video forense. Más adelante se daŕa una vision general de ellos.

6.5 Estructura

El resto de este trabajo se divide en 5 capítulos más:

Capítulo 2 comienza con una introducción sobre cómo funcionan los archivos de video, y continua con una visión general sobre el análisis forense de video. A partir de los conceptos principales, se detallan algunas las técnicas desarrolladas para cubrir sus tres áreas principales: Adquisición, Compresión y Compresión de video. Sin embargo, ninguna de estas técnicas está relacionada con los contenedores de video. A partir de este punto se centra en el estado del arte del análisis de contenedores de video. El trabajo de [ISF+19] es el tema principal de este capítulo, ya que es uno de los pocos trabajos sobre el tema, y el más reciente y avanzado. Este trabajo es analizado y será la base del próximo capítulo.

El capítulo 3 contiene las propuestas del trabajo. En el se explica como reformulando la base del trabajo [ISF+19] se pueden aplicar tecnicas de agrupamiento. Además detalla las técnicas utilizadas. Un paso importanté es la elección de la representacion de los video. Esas diferentes formas de abordar los datos son el núcleo de este trabajo, ya que la representación adecuada de los datos es crucial para la agrupación sin supervisión. Para finalizar, se proponen dos algoritmos para realizar el agrupamiento.

El capítulo 4 describe los experimentos realizados y sus resultados. Se divide en dos fases. Comienza con un análisis del conjunto de datos en diferentes representaciones distintas. Para ello mediremos el coeficiente de silueta de las diferentes representaciones. Para finalizar, los dos algoritmos de agrupación se ponen a prueba utilizando las representaciones más prometedoras.

El Capítulo 5 concluye este trabajo, lo resume y define las posibles mejoras o ampliaciones que podrían realizarse en un futuro.

Chapter 7

Conclusiones y trabajo futuro

Este capítulo resume los hallazgos y logros de esta trabajo y establece líneas de acción para futuros trabajos.

7.1 Conclusiones

A lo largo de este documento, se ha mostrado cómo la estructura del archivo de video puede ser explotada para agrupar videos por fuente de datos, sin necesidad de entrenar un clasificador. Se comprobó que la parte más importante es el correcto procesamiento de los datos antes del agrupamiento. Como se muestra en la sección 4.3, la correcta preparación previa de los datos otorga la separación necesaria necesaria para el agrupamiento. Este mismo criterio también es aplicable a cualquier técnica futura que pretenda extraer conclusiones del análisis de archivos de video.

El uso de algoritmos simples también demostró ser eficaz al separar los archivos de video por marca. Los resultados fueron positivos y se demostró que un algoritmo puede agrupar correctamente los videos en grupos homogéneos por marca, incluso si aparecen muchos grupos. Otro algoritmo también presentaba una forma de mejorar el agrupamiento en términos generales, incluso si la propiedad interesante de homogeneidad.

7.2 Trabajo Futuro

En la misma dirección establecida por este trabajo, surgen algunas líneas de desarrollo.

- **Combinar elementos**
  Las representaciones de los videos pueden ser combinadas, buscando representaciones alternativas que combinen fortalezas. Por ejemplo, combinar PathOrderField y PathOrderFieldValue usando el primero solo para los campos que se espera que tengan valores diferentes. O incluso desarrollando métricas que combinen múltiples representaciones.

- **Aumentar el conjunto de datos**
  El conjunto de datos actual es muy limitado. La fuerza de la agrupación no supervisada está en la capacidad de lidiar con elementos invisibles. Puede ser útil aumentar el tamaño del conjunto de datos, tanto en la cantidad de videos por dispositivo como en los dispositivos en sí, para validar los resultados obtenidos.

- **Validación cruzada para dispositivos desconocidos**
  Si se amplía el conjunto de datos, se puede usar la validación cruzada para aumentar

la confiabilidad de los resultados eliminando una de las clases a cada vez. Es importante eliminar completamente el dispositivo, para simular que el dispositivo sea completamente nuevo.

- **Validación de integridad o clasificación**
  Las representaciones definidas de videos pueden ser adoptadas para otros problemas. Por ejemplo, clasificación, o validación de integridad. No hay ninguna razón por la que los enfoques de representación de video no puedan ser válidos para ello.

Appendices

Appendix A

Results of Silhouette Coefficient

Dataset	Class	Filter	Metric	Full	Original
PathOrderFieldValue	Brand	No	cosine	0,53515	0,368837
PathOrderFieldValue	Brand	No	dice	0,534419	0,367728
PathOrderFieldValue	Brand	No	dissimilarity	0,536545	0,370083
PathOrderFieldValue	Brand	No	euclidean	0,331332	0,284307
PathOrderFieldValue	Brand	No	hamming	0,523232	0,364287
PathOrderFieldValue	Brand	No	jaccard	0,47348	0,372615
PathOrderFieldValue	Brand	No	sokalsneath	0,388865	0,368889
PathOrderFieldValue	Brand	No	yule	0,267601	0,414133
PathOrderFieldValue	Brand	Yes	cosine	0,586047	0,625032
PathOrderFieldValue	Brand	Yes	dice	0,584364	0,625443
PathOrderFieldValue	Brand	Yes	dissimilarity	0,588605	0,624515
PathOrderFieldValue	Brand	Yes	euclidean	0,38382	0,480236
PathOrderFieldValue	Brand	Yes	hamming	0,573141	0,611083
PathOrderFieldValue	Brand	Yes	jaccard	0,544168	0,609646
PathOrderFieldValue	Brand	Yes	sokalsneath	0,482091	0,574963
PathOrderFieldValue	Brand	Yes	yule	0,342589	0,530519
PathOrderFieldValue	Device	No	cosine	0,470177	0,29106
PathOrderFieldValue	Device	No	dice	0,470026	0,291339
PathOrderFieldValue	Device	No	dissimilarity	0,470983	0,290757
PathOrderFieldValue	Device	No	euclidean	0,29627	0,201258
PathOrderFieldValue	Device	No	hamming	0,473966	0,293166
PathOrderFieldValue	Device	No	jaccard	0,426687	0,276334

PathOrderFieldValue	Device	No	sokalsneath	0,358566	0,243252
PathOrderFieldValue	Device	No	yule	0,288609	0,385987
PathOrderFieldValue	Device	Yes	cosine	0,537889	0,250566
PathOrderFieldValue	Device	Yes	dice	0,536964	0,249802
PathOrderFieldValue	Device	Yes	dissimilarity	0,539669	0,251239
PathOrderFieldValue	Device	Yes	euclidean	0,354567	0,180489
PathOrderFieldValue	Device	Yes	hamming	0,539427	0,2464
PathOrderFieldValue	Device	Yes	jaccard	0,512931	0,240534
PathOrderFieldValue	Device	Yes	sokalsneath	0,467377	0,216695
PathOrderFieldValue	Device	Yes	yule	0,39137	0,300273
PathOrderFieldValue	Model	No	cosine	0,430461	0,494269
PathOrderFieldValue	Model	No	dice	0,429383	0,495649
PathOrderFieldValue	Model	No	dissimilarity	0,432278	0,492734
PathOrderFieldValue	Model	No	euclidean	0,273198	0,33593
PathOrderFieldValue	Model	No	hamming	0,430197	0,479556
PathOrderFieldValue	Model	No	jaccard	0,391089	0,44988
PathOrderFieldValue	Model	No	sokalsneath	0,330802	0,381149
PathOrderFieldValue	Model	No	yule	0,225353	0,46679
PathOrderFieldValue	Model	Yes	cosine	0,485664	0,416962
PathOrderFieldValue	Model	Yes	dice	0,483662	0,41707
PathOrderFieldValue	Model	Yes	dissimilarity	0,488659	0,416838
PathOrderFieldValue	Model	Yes	euclidean	0,323058	0,305293
PathOrderFieldValue	Model	Yes	hamming	0,483099	0,419467
PathOrderFieldValue	Model	Yes	jaccard	0,462808	0,418572
PathOrderFieldValue	Model	Yes	sokalsneath	0,423987	0,403479
PathOrderFieldValue	Model	Yes	yule	0,314573	0,517986
PathOrderField	Brand	No	cosine	0,760865	0,20602
PathOrderField	Brand	No	dice	0,75796	0,205827
PathOrderField	Brand	No	dissimilarity	0,765278	0,20625
PathOrderField	Brand	No	euclidean	0,758168	0,239737
PathOrderField	Brand	No	hamming	0,754566	0,205161

PathOrderField	Brand	No	jaccard	0,76165	0,210858
PathOrderField	Brand	No	sokalsneath	0,757202	0,220384
PathOrderField	Brand	No	yule	0,367526	-0,019762
PathOrderField	Brand	Yes	cosine	0,725104	0,617741
PathOrderField	Brand	Yes	dice	0,720333	0,617422
PathOrderField	Brand	Yes	dissimilarity	0,731966	0,618083
PathOrderField	Brand	Yes	euclidean	0,747372	0,67382
PathOrderField	Brand	Yes	hamming	0,719411	0,605323
PathOrderField	Brand	Yes	jaccard	0,730547	0,647518
PathOrderField	Brand	Yes	sokalsneath	0,732937	0,661751
PathOrderField	Brand	Yes	yule	0,215222	0,358059
PathOrderField	Device	No	cosine	0,669923	0,219734
PathOrderField	Device	No	dice	0,667933	0,21973
PathOrderField	Device	No	dissimilarity	0,673406	0,219739
PathOrderField	Device	No	euclidean	0,686581	0,239942
PathOrderField	Device	No	hamming	0,675333	0,219474
PathOrderField	Device	No	jaccard	0,674612	0,224072
PathOrderField	Device	No	sokalsneath	0,680664	0,23195
PathOrderField	Device	No	yule	0,253943	0,085106
PathOrderField	Device	Yes	cosine	0,625299	0,228964
PathOrderField	Device	Yes	dice	0,621511	0,228779
PathOrderField	Device	Yes	dissimilarity	0,631153	0,229179
PathOrderField	Device	Yes	euclidean	0,666356	0,259924
PathOrderField	Device	Yes	hamming	0,631341	0,228002
PathOrderField	Device	Yes	jaccard	0,634604	0,233376
PathOrderField	Device	Yes	sokalsneath	0,648029	0,242164
PathOrderField	Device	Yes	yule	0,093038	0,055203
PathOrderField	Model	No	cosine	0,607844	0,612844
PathOrderField	Model	No	dice	0,605562	0,612535
PathOrderField	Model	No	dissimilarity	0,611665	0,613171
PathOrderField	Model	No	euclidean	0,630161	0,665327

PathOrderField	Model	No	hamming	0,612583	0,600301
PathOrderField	Model	No	jaccard	0,611593	0,64121
PathOrderField	Model	No	sokalsneath	0,618012	0,655627
PathOrderField	Model	No	yule	0,205808	0,531935
PathOrderField	Model	Yes	cosine	0,563254	0,20292
PathOrderField	Model	Yes	dice	0,559121	0,202918
PathOrderField	Model	Yes	dissimilarity	0,569513	0,202922
PathOrderField	Model	Yes	euclidean	0,609536	0,226282
PathOrderField	Model	Yes	hamming	0,568651	0,202732
PathOrderField	Model	Yes	jaccard	0,57137	0,207868
PathOrderField	Model	Yes	sokalsneath	0,584939	0,216782
PathOrderField	Model	Yes	yule	0,042782	0,028019
PathFieldValue	Brand	No	cosine	0,441874	0,421472
PathFieldValue	Brand	No	dice	0,442237	0,421195
PathFieldValue	Brand	No	dissimilarity	0,442118	0,421756
PathFieldValue	Brand	No	euclidean	0,259759	0,284369
PathFieldValue	Brand	No	hamming	0,429031	0,42132
PathFieldValue	Brand	No	jaccard	0,392508	0,40911
PathFieldValue	Brand	No	sokalsneath	0,321512	0,386683
PathFieldValue	Brand	No	yule	0,653203	0,559167
PathFieldValue	Brand	Yes	cosine	0,490424	0,55874
PathFieldValue	Brand	Yes	dice	0,490501	0,559707
PathFieldValue	Brand	Yes	dissimilarity	0,49108	0,557682
PathFieldValue	Brand	Yes	euclidean	0,307303	0,383603
PathFieldValue	Brand	Yes	hamming	0,480587	0,556437
PathFieldValue	Brand	Yes	jaccard	0,456325	0,539457
PathFieldValue	Brand	Yes	sokalsneath	0,402169	0,50485
PathFieldValue	Brand	Yes	yule	0,661963	0,686108
PathFieldValue	Device	No	cosine	0,426858	0,282706
PathFieldValue	Device	No	dice	0,426852	0,283189
PathFieldValue	Device	No	dissimilarity	0,427514	0,282209

PathFieldValue	Device	No	euclidean	0,251006	0,180245
PathFieldValue	Device	No	hamming	0,421543	0,284848
PathFieldValue	Device	No	jaccard	0,382365	0,260756
PathFieldValue	Device	No	sokalsneath	0,316657	0,226769
PathFieldValue	Device	No	yule	0,655339	0,449343
PathFieldValue	Device	Yes	cosine	0,492191	0,271416
PathFieldValue	Device	Yes	dice	0,491913	0,27159
PathFieldValue	Device	Yes	dissimilarity	0,493339	0,271226
PathFieldValue	Device	Yes	euclidean	0,304278	0,174817
PathFieldValue	Device	Yes	hamming	0,487856	0,272178
PathFieldValue	Device	Yes	jaccard	0,462684	0,250766
PathFieldValue	Device	Yes	sokalsneath	0,413705	0,218741
PathFieldValue	Device	Yes	yule	0,696647	0,417425
PathFieldValue	Model	No	cosine	0,407695	0,376446
PathFieldValue	Model	No	dice	0,407434	0,37767
PathFieldValue	Model	No	dissimilarity	0,408609	0,375146
PathFieldValue	Model	No	euclidean	0,24136	0,238448
PathFieldValue	Model	No	hamming	0,401229	0,373635
PathFieldValue	Model	No	jaccard	0,36561	0,342924
PathFieldValue	Model	No	sokalsneath	0,303518	0,292206
PathFieldValue	Model	No	yule	0,615433	0,555977
PathFieldValue	Model	Yes	cosine	0,466047	0,423723
PathFieldValue	Model	Yes	dice	0,465294	0,424046
PathFieldValue	Model	Yes	dissimilarity	0,467682	0,42339
PathFieldValue	Model	Yes	euclidean	0,291929	0,278801
PathFieldValue	Model	Yes	hamming	0,460221	0,425636
PathFieldValue	Model	Yes	jaccard	0,438799	0,409287
PathFieldValue	Model	Yes	sokalsneath	0,394033	0,383084
PathFieldValue	Model	Yes	yule	0,647963	0,589758
PathField	Brand	No	cosine	0,711387	0,036064
PathField	Brand	No	dice	0,709286	0,036025

PathField	Brand	No	dissimilarity	0,719096	0,036105
PathField	Brand	No	euclidean	0,726358	0,045549
PathField	Brand	No	hamming	0,713778	0,03582
PathField	Brand	No	jaccard	0,717522	0,036224
PathField	Brand	No	sokalsneath	0,720447	0,036579
PathField	Brand	No	yule	0,682742	-0,103007
PathField	Brand	Yes	cosine	0,010912	0,355599
PathField	Brand	Yes	dice	0,010528	0,355274
PathField	Brand	Yes	dissimilarity	0,011245	0,355875
PathField	Brand	Yes	euclidean	0,042902	0,398712
PathField	Brand	Yes	hamming	0,006102	0,35239
PathField	Brand	Yes	jaccard	0,010299	0,356251
PathField	Brand	Yes	sokalsneath	0,010344	0,358512
PathField	Brand	Yes	yule	0,250225	-0,011771
PathField	Device	No	cosine	0,649546	0,223533
PathField	Device	No	dice	0,647173	0,223479
PathField	Device	No	dissimilarity	0,657656	0,22359
PathField	Device	No	euclidean	0,668669	0,221577
PathField	Device	No	hamming	0,656197	0,223113
PathField	Device	No	jaccard	0,656811	0,223448
PathField	Device	No	sokalsneath	0,661949	0,223388
PathField	Device	No	yule	0,601839	0,000415
PathField	Device	Yes	cosine	-0,050331	0,270904
PathField	Device	Yes	dice	-0,050631	0,27079
PathField	Device	Yes	dissimilarity	-0,049993	0,271023
PathField	Device	Yes	euclidean	-0,044316	0,280359
PathField	Device	Yes	hamming	-0,05114	0,270286
PathField	Device	Yes	jaccard	-0,049942	0,270985
PathField	Device	Yes	sokalsneath	-0,048827	0,271344
PathField	Device	Yes	yule	-0,33778	-0,018232
PathField	Model	No	cosine	0,597117	0,642781

PathField	Model	No	dice	0,594699	0,642565
PathField	Model	No	dissimilarity	0,605274	0,642909
PathField	Model	No	euclidean	0,622528	0,646994
PathField	Model	No	hamming	0,603564	0,639245
PathField	Model	No	jaccard	0,604256	0,642401
PathField	Model	No	sokalsneath	0,609336	0,642292
PathField	Model	No	yule	0,561265	0,185868
PathField	Model	Yes	cosine	-0,112228	0,086241
PathField	Model	Yes	dice	-0,112586	0,086238
PathField	Model	Yes	dissimilarity	-0,11183	0,086243
PathField	Model	Yes	euclidean	-0,104571	0,086335
PathField	Model	Yes	hamming	-0,113157	0,086214
PathField	Model	Yes	jaccard	-0,112109	0,086258
PathField	Model	Yes	sokalsneath	-0,111276	0,08629
PathField	Model	Yes	yule	-0,364412	-0,056916

Table A.1: Silhouette Coefficient of the full dataset, and original videos only, for multiple configurations

Appendix B

Results of DBSCAN

Elements	Label	Filter	Metric	threshold	Size train	AMI train	Vmeasure traim	Homogenity traim	Completness traim	Size test	AMI test	Vmeasure test	Homogenity test	Completness test
PathOrderField	Model	FALSE	euclidean	0,1	27	0,8917380342	0,9021731049	0,9021997069	0,9021465044	24	0,8781267676	0,8961167813	0,8967777095	0,8954568267
PathOrderField	Model	FALSE	sokalsneath	1,00E-05	27	0,8917380342	0,9021731049	0,9021997069	0,9021465044	24	0,8781267676	0,8961167813	0,8967777095	0,8954568267
PathOrderField	Model	FALSE	jaccard	1,00E-05	27	0,8917380342	0,9021731049	0,9021997069	0,9021465044	24	0,8781267676	0,8961167813	0,8967777095	0,8954568267
PathField	Model	FALSE	euclidean	0,1	26	0,8814711879	0,8968454845	0,8925405389	0,9011921591	23	0,8680131861	0,8903262939	0,8863336336	0,8943550884
PathField	Model	FALSE	sokalsneath	1,00E-05	26	0,8814711879	0,8968454845	0,8925405389	0,9011921591	23	0,8680131861	0,8903262939	0,8863336336	0,8943550884
PathField	Model	FALSE	dissimilarity	1,00E-05	26	0,8814711879	0,8968454845	0,8925405389	0,9011921591	23	0,8680131861	0,8903262939	0,8863336336	0,8943550884
PathFieldValue	Brand	TRUE	yule	0,0007718807334	14	0,8086576767	0,8219536847	0,8299215379	0,8141373709	24	0,6240358894	0,7526446509	0,8953194854	0,6491918198
PathFieldValue	Brand	TRUE	dice	0,1037681937	11	0,8066028984	0,8173397848	0,8111926075	0,8235808395	11	0,8127686271	0,8297444104	0,8383557252	0,8213082022
PathFieldValue	Brand	TRUE	cosine	0,1037681937	11	0,8066028984	0,8173397848	0,8111926075	0,8235808395	11	0,8127686271	0,8297444104	0,8383557252	0,8213082022
PathOrderField	Brand	TRUE	euclidean	3,874797474	16	0,8209141133	0,8702636465	0,9188945013	0,8265214698	14	0,8097760549	0,86398897	0,9134341435	0,8196219514
PathOrderField	Brand	TRUE	sokalsneath	0,1502127605	16	0,8209141133	0,8702636465	0,9188945013	0,8265214698	14	0,8097760549	0,86398897	0,9134341435	0,8196219514
PathOrderField	Brand	TRUE	jaccard	0,07862892341	16	0,8209141133	0,8702636465	0,9188945013	0,8265214698	14	0,8097760549	0,86398897	0,9134341435	0,8196219514

Table B.1: Results of looking for the adequate ϵ for the full dataset with partition of 66% for train and 33% for test.

Elements	Label	Filter	Metric	threshold	Size train	AMI train	Vmeasure traim	Homogenity traim	Completness traim	Size test	AMI test	Vmeasure test	Homogenity test	Completness test
PathOrderField	Brand	TRUE	euclidean	2,243882923	18	0,8644169415	0,8980196962	0,9312736086	0,867058767	13	0,6476037738	0,7767827312	0,8925518745	0,6875973729
PathOrderField	Brand	TRUE	sokalsneath	0,0375229685	18	0,8644169415	0,8980196962	0,9312736086	0,867058767	13	0,6476037738	0,7767827312	0,8925518745	0,6875973729
PathOrderField	Brand	TRUE	jaccard	0,01964141133	18	0,8644169415	0,8980196962	0,9312736086	0,867058767	13	0,6476037738	0,7767827312	0,8925518745	0,6875973729
PathOrderField	Model	FALSE	euclidean	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	20	0,7170284199	0,8620895003	0,8101559335	0,9211373268
PathOrderField	Model	FALSE	sokalsneath	1,00E-05	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	20	0,7170284199	0,8620895003	0,8101559335	0,9211373268
PathOrderField	Model	FALSE	jaccard	1,00E-05	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	20	0,7170284199	0,8620895003	0,8101559335	0,9211373268
PathFieldValue	Brand	TRUE	yule	0,0004641588834	26	0,8649806126	0,9298375507	1	0,8688751426	14	0,6218089143	0,6993416544	0,7292722503	0,671771009
PathFieldValue	Brand	TRUE	dice	0,05688792333	29	0,8418285502	0,9170934554	1	0,8468814414	22	0,4972411581	0,7042159845	0,9223701703	0,5695168315
PathFieldValue	Brand	TRUE	cosine	0,05688792333	29	0,8418285502	0,9170934554	1	0,8468814414	22	0,4972411581	0,7042159845	0,9223701703	0,5695168315
PathField	Model	FALSE	euclidean	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	19	0,694938998	0,8505956101	0,7913610148	0,9194152639
PathField	Model	FALSE	sokalsneath	1,00E-05	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	19	0,694938998	0,8505956101	0,7913610148	0,9194152639
PathField	Model	FALSE	dissimilarity	1,00E-05	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	19	0,694938998	0,8505956101	0,7913610148	0,9194152639

Table B.2: Results of looking for the adequate ϵ for the original videos in the dataset with partition of 66% for train (oversampled) and 33% for test.

Appendix C

Results of Hierarchical Clustering

Elements	Label	Filter	Metric	linkage	threshold	Size_full	AMI train	Vmeasure train	Homogenity train	Completness train	Size test	AMI test	Vmeasure test	Homogenity test	Completness test
PathOrderField	Brand	TRUE	euclidean	single	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	euclidean	complete	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	euclidean	average	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	euclidean	weighted	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	sokalsneath	single	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	sokalsneath	complete	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	sokalsneath	average	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	sokalsneath	weighted	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	jaccard	single	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	jaccard	complete	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	jaccard	average	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Brand	TRUE	jaccard	weighted	0,1	27	0,7672113227	0,8660872049	0,9779863042	0,7771655827	26	0,7606424703	0,8674586807	0,98061466	0,7777157923
PathOrderField	Model	FALSE	euclidean	single	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	euclidean	complete	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	euclidean	average	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	euclidean	weighted	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	sokalsneath	single	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	sokalsneath	complete	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	sokalsneath	average	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	sokalsneath	weighted	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	jaccard	single	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	jaccard	complete	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	jaccard	average	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathOrderField	Model	FALSE	jaccard	weighted	0,1	28	0,8918086828	0,9029119117	0,9035465971	0,9022781173	31	0,8736400726	0,9035085934	0,9146801084	0,892606674
PathFieldValue	Brand	TRUE	yule	single	1,14879084	303	0,3253678702	0,5858554822	0,9941127098	0,4153015302	145	0,3697587015	0,6310203274	0,9839463691	0,4644348667
PathFieldValue	Brand	TRUE	yule	complete	1,14879084	244	0,2590939357	0,5103760331	1	0,342620718	102	0,3307711478	0,5833533281	1	0,4117846317
PathFieldValue	Brand	TRUE	yule	average	1,14879084	223	0,2743852663	0,5178890615	0,9782954492	0,3521567146	150	0,2609567422	0,5330229499	0,9839463691	0,3655146972
PathFieldValue	Brand	TRUE	yule	weighted	1,14879084	216	0,2765069736	0,52161655	1	0,3528289971	141	0,2710790745	0,5421886301	1	0,3719196058
PathFieldValue	Brand	TRUE	dice	single	1,127739612	514	0,1772634985	0,468861464	1	0,3062175322	151	0,3573685412	0,6255635348	1	0,455141835
PathFieldValue	Brand	TRUE	dice	complete	1,14879084	206	0,2878835172	0,5310735398	0,9909315448	0,3627387054	101	0,3288212357	0,5799541733	0,9929730755	0,4095890221
PathFieldValue	Brand	TRUE	dice	average	1,14879084	261	0,2500306255	0,5025917305	0,991806296	0,336574356	147	0,2714405023	0,5444045959	1	0,3740081855
PathFieldValue	Brand	TRUE	dice	weighted	1,14879084	248	0,2527539959	0,5020310074	0,9833058709	0,3370589448	152	0,2633788962	0,5389134633	1	0,3688443153
PathFieldValue	Brand	TRUE	cosine	single	1,127739612	514	0,1772634985	0,468861464	1	0,3062175322	151	0,3573685412	0,6255635348	1	0,455141835
PathFieldValue	Brand	TRUE	cosine	complete	1,14879084	204	0,2885819094	0,531415036	0,9909315448	0,3630574161	101	0,3288212357	0,5799541733	0,9929730755	0,4095890221
PathFieldValue	Brand	TRUE	cosine	average	1,14879084	245	0,2581354115	0,5073490838	0,991806296	0,3408551474	153	0,264824424	0,540521834	1	0,3703528059
PathFieldValue	Brand	TRUE	cosine	weighted	1,14879084	235	0,2605972384	0,5070546532	0,9833058709	0,3416035051	149	0,2680494753	0,5420163042	1	0,3717574523
PathField	Model	FALSE	euclidean	single	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	euclidean	complete	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	euclidean	average	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	euclidean	weighted	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	sokalsneath	single	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	sokalsneath	complete	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	sokalsneath	average	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	sokalsneath	weighted	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	dissimilarity	single	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	dissimilarity	complete	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	dissimilarity	average	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468
PathField	Model	FALSE	dissimilarity	weighted	0,1	27	0,882859057	0,897591477	0,893887429	0,90132635	30	0,8726395126	0,8978232813	0,9042360325	0,8915008468

Table 1: Results of looking for the adequate inconsistency threshold for the full dataset with partition of 66% for train and 33% for test.

Elements	Label	Filter	Metric	linkage	threshold	Size_full	AMI train	Vmeasure train	Homogenity train	Completness train	Size test	AMI test	Vmeasure test	Homogenity test	Completness test
PathOrderField	Brand	TRUE	euclidean	single	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	euclidean	complete	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	euclidean	average	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	euclidean	weighted	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	sokalsneath	single	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	sokalsneath	complete	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	sokalsneath	average	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	sokalsneath	weighted	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	jaccard	single	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	jaccard	complete	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	jaccard	average	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Brand	TRUE	jaccard	weighted	0,1	22	0,8394065406	0,8851085014	0,9312736086	0,8433042141	21	0,5718463822	0,7554844832	0,9396854295	0,6316632858
PathOrderField	Model	FALSE	euclidean	single	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	euclidean	complete	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	euclidean	average	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	euclidean	weighted	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	sokalsneath	single	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	sokalsneath	complete	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	sokalsneath	average	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	sokalsneath	weighted	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	jaccard	single	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	jaccard	complete	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	jaccard	average	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathOrderField	Model	FALSE	jaccard	weighted	0,1	25	0,7954332156	0,8624530073	0,8182045255	0,9117610222	24	0,7392562074	0,8710043845	0,8316626412	0,9142530618
PathFieldValue	Brand	TRUE	yule	single	1,14879084	249	0,3883109338	0,6362897978	1	0,4665872535	48	0,3849381779	0,659405752	0,9551710435	0,5034990692
PathFieldValue	Brand	TRUE	yule	complete	1,14879084	209	0,4025091913	0,6429987812	1	0,4738380278	42	0,4037991826	0,6793090234	1	0,5143587981
PathFieldValue	Brand	TRUE	yule	average	1,14879084	202	0,410605319	0,647418245	1	0,4786536877	42	0,4123070073	0,6853124522	1	0,5212740118
PathFieldValue	Brand	TRUE	yule	weighted	1,14879084	212	0,4024445483	0,6430262622	1	0,4738678755	43	0,4096934889	0,6841534763	1	0,5199340987
PathFieldValue	Brand	TRUE	dice	single	1,127739612	247	0,3889186365	0,6365477404	1	0,4668647075	51	0,3806830545	0,6692646712	1	0,5029284612
PathFieldValue	Brand	TRUE	dice	complete	1,14879084	208	0,4027882446	0,6431116404	1	0,4739606142	40	0,3960781483	0,6636707892	0,9655462209	0,5055971871
PathFieldValue	Brand	TRUE	dice	average	1,14879084	198	0,4132766477	0,6490865071	1	0,480479698	42	0,4339877868	0,6998975424	1	0,5383402964
PathFieldValue	Brand	TRUE	dice	weighted	1,14879084	208	0,4036484522	0,6435449659	1	0,4744314775	40	0,4419582295	0,7037669303	1	0,5429323991
PathFieldValue	Brand	TRUE	cosine	single	1,127739612	247	0,3889186365	0,6365477404	1	0,4668647075	51	0,3806830545	0,6692646712	1	0,5029284612
PathFieldValue	Brand	TRUE	cosine	complete	1,14879084	208	0,4027882446	0,6431116404	1	0,4739606142	40	0,3960781483	0,6636707892	0,9655462209	0,5055971871
PathFieldValue	Brand	TRUE	cosine	average	1,14879084	198	0,4132766477	0,6490865071	1	0,480479698	42	0,4339877868	0,6998975424	1	0,5383402964
PathFieldValue	Brand	TRUE	cosine	weighted	1,14879084	208	0,4036484522	0,6435449659	1	0,4744314775	40	0,4419582295	0,7037669303	1	0,5429323991
PathField	Model	FALSE	euclidean	single	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	euclidean	complete	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	euclidean	average	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	euclidean	weighted	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	sokalsneath	single	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	sokalsneath	complete	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	sokalsneath	average	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	sokalsneath	weighted	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	dissimilarity	single	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	dissimilarity	complete	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	dissimilarity	average	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332
PathField	Model	FALSE	dissimilarity	weighted	0,1	24	0,7846926658	0,8561388975	0,8077309428	0,9107190085	23	0,7158788326	0,8597823562	0,8128677226	0,9124440332

Table 2: Results of looking for the adequate inconsistency threshold for the original videos in the dataset with partition of 66% for train (oversampled) and 33% for test.

Bibliography

[ACDM+17] I. Amerini, R. Caldelli, A. Del Mastio, A. Di Fuccia, C. Molinari, and A. P. Rizzo. Dealing with video source identification in social networks. *Signal Processing: Image Communication*, 57:1–7, 2017.

[Ale] Alexa. The top 500 sites on the web. https://www.alexa.com/topsites.

[APA18] APACHE. https://github.com/sannies/mp4parser, Jul 2018.

[App16] Apple. QuickTime File Format Specification: Overview. https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap1/qtff1.html, Sep 2016.

[Asl18] S. Aslam. YouTube by the Numbers: Stats, Demographics Fun Facts. =https://www.omnicoreagency.com/youtube-statistics/, Dec 2018.

[Bec15] M Beck. Reversal of Facebook: photo posts now drive lowest organic reach, Feb 2015.

[BS15] D. J. Balding and C. D. Steele. *Weight-of-evidednce for forensic DNA profiles*. John Wiley Sons, May 2015.

[Cis18] Cisco. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, Nov 2018.

[COF12] V. Conotter, J.F. O'Brien, and H. Farid. Exposing digital forgeries in ballistic motion. *Institute of Electrical and Electronics Engineers Transactions on Information Forensics and Security*, 7(1):283–296, 2012.

[EKSX96] M. Ester, H. Kriegel, J. Sander, and X. Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, August 1996.

[FDQ00] Z. Fan and R. De Queiroz. Maximum likelihood estimation of JPEG quantization table in the identification of bitmap compression history. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 1, pages 948–951, September 2000.

[FDQ03] Z. Fan and R. L. De Queiroz. Identification of bitmap compression history: JPEG detection and quantizer estimation. *Institute of Electrical and Electronics Engineers Transactions on Image Processing*, 12(2):230–235, April 2003.

[FSS07] Dongdong Fu, Yun Q Shi, and Wei Su. A generalized Benford's law for JPEG coefficients and its applications in image forensics. In *Security, Steganography, and Watermarking of Multimedia Contents nine*, volume 6505, San Jose, CA, United States, February 2007.

[GFK14] T. Gloe, A. Fischer, and M. Kirchner. Forensic analysis of video file formats. *Digital Investigation*, 11:68–76, 2014.

[HHLH08] Chih-Chung Hsu, Tzu-Yi Hung, Chia-Wen Lin, and Chiou-Ting Hsu. Video forgery detection using correlation of noise residue. In *Institute of Electrical and Electronics Engineers tenth workshop on Multimedia Signal Processing*, pages 170–174, November 2008.

[HLWT06] Junfeng He, Zhouchen Lin, Lifeng Wang, and Xiaoou Tang. Detecting doctored JPEG images via DCT coefficient analysis. In *European conference on computer vision*, volume 3953, pages 423–435, Graz, Austria, May 2006.

[Int15] International Organization for Standardization. ISO/IEC 14496-2:2004: Coding of audio-visual objects – Part 2: Visual. https://www.iso.org/standard/39259.html, August 2015.

[Int17] International Organization for Standardization. ISO/IEC 14496-12:2015: Coding of audio-visual objects – Part 12: ISO base media file format. https://www.iso.org/standard/68960.html, Aug 2017.

[ISF+19] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva. A Video Forensic Framework for the Unsupervised Analysis of MP4-Like File Container. *Institute of Electrical and Electronics Engineers Transactions on Information Forensics and Security*, 14(3):635–645, 2019.

[MCP+07] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini. Detection of malevolent changes in digital video for forensic applications. In *Security, steganography, and watermarking of multimedia contents nine*, volume 6505, San Jose, CA, United States, January 2007.

[MFB+12] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro. An overview on video forensics. *Asia-Pacific Signal and Information Processing Association Transactions on Signal and Information Processing*, 1:1229–1233, 2012.

[MKR99] M. K. Mihcak, I. Kozintsev, and K. Ramchandran. Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 6, pages 3253–3256, March 1999.

[nas92] The effects of video compression on acceptability of images for monitoring life sciences experiments. Technical Report, National Aeronautics and Space Administration, July 1992.

[oDE16] Scientific Working Group on Digital Evidence. SWGDE Digital Multimedia Evidence Glossary. https://www.swgde.org/documents/Current%20Documents/SWGDE%20Digital%20and%20Multimedia%20Evidence%20Glossary, June 2016.

[RVBV16] S. Romano, N. X. Vinh, J. Bailey, and K. Verspoor. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17:4635–4666, 2016.

[Sci17] Scientific Working Group on Digital Evidence. Swgde technical overview of digital video files. https://www.swgde.org/documents/Current%20Documents/SWGDE%20Technical%20Overview%20of%20Digital%20Video%20Files, August 2017.

[Sci18] Scientific Working Group on Digital Evidence. Swgde best practices for image authentication. https://www.swgde.org/documents/Current%20Documents/SWGDE%20Best%20Practices%20for%20Digital%20Forensic%20Video%20Analysis, August 2018.

[SPC17] D. Shullani, A. Piva, and L. Chisci. Video forensic tools exploiting features from video-container to video-encoder level. PhD Thesis, Dipartamento di Ingenieria dell'Informazione, Universita Degli Studi Firenze, Italia, April 2017.

[VEB10] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, January 2010.

[vis17] VISION: a video and image dataset for source identification. *European Association for Signal Processing Journal on Information Security*, 2017(1):150, Oct 2017.

[VTT10] G. Valenzise, M. Tagliasacchi, and S. Tubaro. Estimating QP and motion vectors in H. 264/AVC video from decoded pixels. In *Proceedings of the second Association for Computing Machinery workshop on Multimedia in forensics, security and intelligence*, pages 89–92, 978-1-4503-0157-2, October 2010.

[WF06] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double MPEG compression. In *Proceedings of the eighth Association for Computing Machinery workshop on Multimedia and security*, pages 37–47, Geneva, Switzerland, September 2006.

[WF09] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double quantization. In *Proceedings of the eleventh Association for Computing Machinery workshop on Multimedia and security*, pages 39–48, Princeton, NJ, United States, September 2009.

[ZSZ09] J. Zhang, Y. Su, and M. Zhang. Exposing digital video forgery by ghost shadow artifact. In *Proceedings of the First Association for Computing Machinery workshop on Multimedia in forensics*, pages 49–54, October 2009.